Modeling working memory: a computational implementation of the Time-Based Resource-Sharing theory

Oberauer, Klaus ; Lewandowsky, S

Abstract: Working memory is a core concept in cognition, predicting about 50% of the variance in IQ and reasoning tasks. A popular test of working memory is the complex span task, in which encoding of memoranda alternates with processing of distractors. A recent model of complex span performance, the Time-Based-Resource-Sharing (TBRS) model of Barrouillet and colleagues, has seemingly accounted for several crucial findings, in particular the intricate trade-off between deterioration and restoration of memory in the complex span task. According to the TBRS, memory traces decay during processing of the distractors, and they are restored by attentional refreshing during brief pauses in between processing steps. However, to date, the theory has been formulated only at a verbal level, which renders it difficult to test and to be certain of its intuited predictions. We present a computational instantiation of the TBRS and show that it can handle most of the findings on which the verbal model was based. We also show that there are potential challenges to the model that await future resolution. This instantiated model, TBRS*, is the first comprehensive computational model of performance in the complex span paradigm. The Matlab model code is available as a supplementary material of this article.

DOI: https://doi.org/10.3758/s13423-010-0020-6

Psychon Bull Rev (2011) 18:10–45
DOI 10.3758/s13423-010-0020-6

Modeling working memory: a computational implementation of the Time-Based Resource-Sharing theory

Klaus Oberauer · Stephan Lewandowsky

Published online: 16 November 2010

Abstract Working memory is a core concept in cognition, predicting about 50% of the variance in IQ and reasoning tasks. A popular test of working memory is the complex span task, in which encoding of memoranda alternates with processing of distractors. A recent model of complex span performance, the Time-Based-Resource-Sharing (TBRS) model of Barrouillet and colleagues, has seemingly accounted for several crucial findings, in particular the intricate trade-off between deterioration and restoration of memory in the complex span task. According to the TBRS, memory traces decay during processing of the distractors, and they are restored by attentional refreshing during brief pauses in between processing steps. However, to date, the theory has been formulated only at a verbal level, which renders it difficult to test and to be certain of its intuited predictions. We present a computational instantiation of the TBRS and show that it can handle most of the findings on which the verbal model was based. We also show that there are potential challenges to the model that await future resolution. This instantiated model, TBRS*, is the first comprehensive computational model of performance in the complex span paradigm. The Matlab model code is available as a supplementary material of this article.

Electronic supplementary material The online version of this article (doi:10.3758/s13423-010-0020-6) contains supplementary material, which is available to authorized users.

K. Oberauer
Department of Experimental Psychology, University of Bristol, Bristol, UK

K. Oberauer (✉)
Department of Psychology, University of Zurich, Binzmühlestrasse 14/22, 8050 Zurich, Switzerland
e-mail: k.oberauer@psychologie.uzh.ch

S. Lewandowsky
University of Western Australia, Perth, Australia

Keywords Working memory · Computational modeling

Working memory has often been characterized as a system for the simultaneous maintenance and processing of information. This working definition is reflected in the complex span paradigm, which has become the most popular method for psychometric measurement of working memory capacity (Conway et al., 2005) and which also serves in many experimental investigations of working memory processes (e.g., Barrouillet, Bernardin, & Camos, 2004; Friedman & Miyake, 2004; Towse, Hitch, & Hutton, 2000). The complex span paradigm is a generalization of the reading span task (Daneman & Carpenter, 1980). In reading span, participants read a series of sentences and try to remember the last word of each sentence. The number of sentences in each series is gradually increased, and a participant's span is determined as the maximum number of sentence-final words they can recall correctly in the order of presentation on at least 50% of trials. Thus, people alternate between a processing task (i.e., reading the sentences, often accompanied by a judgment about each sentence) and encoding items for later recall (i.e., encoding each sentence's last word). Other researchers have developed arithmetic variants of the task and versions in which the to-be-remembered items are separated from the to-be-processed material (Turner & Engle, 1989). For instance, participants alternate between reading a sentence and encoding a letter for later recall, or between verifying an arithmetic equation and encoding a word.

Reading span and its relatives have turned out to be good predictors of complex cognitive performance such as text comprehension and reasoning; their predictive power is typically larger than that of comparable *simple span* tasks, which ask for immediate serial recall of a list without interspersed processing demand (Ackerman, Beier, &

Boyle, 2005; Conway, Kane, & Engle, 2003; Daneman & Merikle, 1996). The importance of working memory is underscored by the fact that there is a strong relationship between general fluid intelligence (Gf, often measured by Raven's Progressive Matrices test, Raven, Court, & Raven, 1977) and performance in complex span tasks, as well as other tasks that measure working memory capacity (WMC). In an extensive review of 14 data sets, Kane, Hambrick, and Conway (2005) found the correlation between WMC and Gf to be .72—that is, general intelligence shares 50% of the variance with people's ability to perform a fairly straightforward memory task. By implication, a better understanding of complex span performance may yield an insight into the very core of cognition, viz. intelligence.

Further work has shown that the nature and the difficulty of the processing task has no bearing on the validity of complex span tasks as indicators of working memory capacity (Conway & Engle, 1996; Lepine, Barrouillet, & Camos, 2005). Rather, the validity of span tasks increases with the degree of experimental control over people's strategies (Turley-Ames & Whitfield, 2003) and over how much time they assign to the processing component (Friedman & Miyake, 2004; Lepine et al., 2005).

Barrouillet, Camos, and their colleagues have developed the complex span paradigm into a tool that affords better control over people's cognitive processes than the original reading span task (Barrouillet et al., 2004; Barrouillet, Bernardin, Portrat, Vergauwe, & Camos, 2007; Barrouillet & Camos, 2001). In their variant of complex span, the task is entirely computer paced. Presentation of each memory item (e.g., a letter) for a fixed time is followed by a brief processing period, which we will refer to as a *burst*. The processing task is broken down into a number of steps, each of which is prompted by a separate stimulus for a fixed time (we will refer to this stimulus as a *distractor*). For instance, participants are given an initial digit, followed by a series of arithmetic updating instructions (e.g., "+3", "-2"). Participants must respond to each instruction by verbalizing the intermediate result within the allotted processing time. After several such processing steps, the next memorandum is presented, followed by the next series of processing steps. After all the memoranda and processing stimuli have been presented, people are asked to recall the memory items in order without time constraint (see Fig. 1 for an illustration of the complex span paradigm).

The complex span task clearly draws on a large number of cognitive processes, ranging from memory encoding and retrieval to task switching and arithmetic processing—or indeed any other type of processing associated with the distractor task. Any explanation of performance in this non-trivial task must therefore also involve a number of components: At the very least, a theory must specify how items are encoded, how they are maintained or forgotten, and how they are retrieved at the end. In addition, a theory must state what happens to memory while the distractor task is carried out. To date, this has presented a formidable theoretical challenge that only a few models have attempted to tackle.

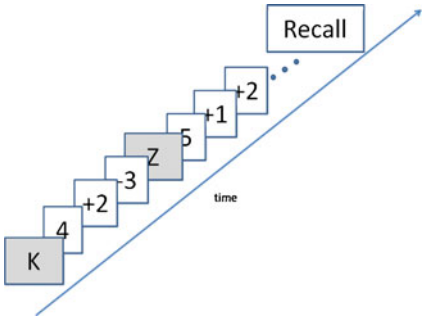

Fig. 1 Schema of the complex span paradigm. Subjects remember the consonants (K, Z), which must be recalled at the end of the trial. Memoranda are separated by a processing task; in this instance, subjects read aloud the arithmetic operations and their running results

Barrouillet et al. (2004) presented a theory of maintenance and processing in working memory that explains a number of findings from their refined version of the complex span paradigm. Their theory, called the time-based resource-sharing (TBRS) model, has proven very successful in accounting for behavior in that paradigm. Unlike other theories of working memory that remain at a more global level (e.g., Cowan, 1995; Kane & Engle, 2000), the TBRS is the first attempt to specify the *processes* underlying complex span performance. Specifically, the TBRS relies on the interplay between two opposing processes, namely temporal decay and compensatory attentional "refreshing" of memory traces. Refreshing competes with other cognitive operations for a general processing bottleneck and serves to restore the integrity of memory traces that inexorably decay during distractor processing. So far, the TBRS has been presented only as a verbal theory, and its predictions are derived by intuiting the effects of decay and refreshing in the complex span task. Like all other verbal theories, the TBRS is thus susceptible to numerous conceptual risks and potential pitfalls that beset purely verbal theorizing (Hintzman, 1991; Lewandowsky, 1993).

This article provides a computational implementation of the TBRS. The purpose of this computational implementation is threefold. First, by implementing the theory as a computational model we make explicit many assumptions about details of the mechanisms of working memory on which the verbal theory is silent. Thus, our computational implementation raises theorizing about working memory to a new level of precision. To date, there exists no detailed and comprehensive computational process model of performance in working-memory tasks. Second, the computational implementation serves as a validity check for the

assumptions underlying the TRBS: The fact that our instantiation of the theory at least approximately reproduces people's behavior reassures us that the assumptions are mutually consistent and viable. Third, the computational model of the TBRS allows us to derive unambiguous quantitative predictions for further empirical tests. In what follows, we first introduce the TBRS theory, then explain our modeling strategy, followed by a presentation of the model itself and an exploration of its behavior.

The time-based resource-sharing theory

The TBRS theory makes the following basic assumptions: Representations in working memory decay over time, but they can be refreshed by directing attention to them. Attention is conceptualized as a domain-general mechanism that can be devoted to only one process at a time, and hence creates a bottleneck. In tasks like the complex span task, the cognitive system must devote attention to carrying out each step of the processing task interleaved with encoding of the memoranda. In between processing steps, however, the attentional mechanism can be used to refresh memory items. Thus, during each processing period the attentional bottleneck is assumed to rapidly switch between carrying out a processing step and refreshing one or more memory items.

Barrouillet and colleagues have condensed these assumptions into a formula for cognitive load, defined as the proportion of time T during which the attentional mechanism is captured by the processing task, and thus the proportion of time during which the memory traces decay without being refreshed. Cognitive load can be expressed by the equation:

$$Cognitive\ Load = t_a N / T,$$

where t_a is the duration for which attention is captured by a processing step, N represents the number of processing steps, and T the total time available for the processing period between two memoranda. If the processing period T is divided equally among all steps, this equation reduces to t_a/t, where t is the presentation time for each individual processing step. Its inverse, $1/t$, is referred to as the *pace* of processing steps.

To illustrate, suppose that memoranda are separated by 2 s, during which people must process two arithmetic steps (e.g., "+1" and "-2") to compute a running total, each of which takes 600 ms. Cognitive load would be equal to 600 × 2 / 2,000 or 0.6. Now suppose the time available is doubled: Cognitive load would be cut in half because 600 × 2 / 4,000 = 0.3. If the number of operations were then doubled in turn, cognitive load would be back to its original value; 600 × 4 / 4,000 = 0.6. These examples illustrate that cognitive load represents the balance between competing effects; viz. the detrimental effect on memory of the processing steps and the memorial restoration afforded during the breaks. The examples also point to a limitation of the testability of the TBRS: Whereas the theory stipulates that cognitive load derives from the duration of *attentional capture* associated with the processing task, it is difficult to measure that duration directly. In most cases, the duration of attentional capture is estimated from the time taken to respond to a distractor stimulus without a memory load (i.e., "offline" in a separate sequence of test trials). Although this indirect measure is satisfactory in most cases, we turn to situations later in which attentional capture might not be fully reflected in overt processing latencies.

Support for the TBRS comes from experiments with the complex span paradigm that have revealed four consistent regularities: (1) The addition of a processing component after encoding of each memory item leads to a substantial drop in recall accuracy, relative to a comparable simple span task in which the memory items are presented uninterrupted. The drop in performance can be large even with a fairly simple and brief processing period – for instance, saying aloud a single word after encoding of each letter can reduce recall accuracy by 20% (Oberauer & Lewandowsky, 2008). (2) Memory accuracy depends on the pace of processing steps; faster paced processing steps lead to worse recall (Barrouillet et al., 2004; Barrouillet et al., 2007). (3) Holding processing pace constant, a more difficult processing task results in worse memory (for an illustration, see Fig. 2). Difficulty of the processing task per

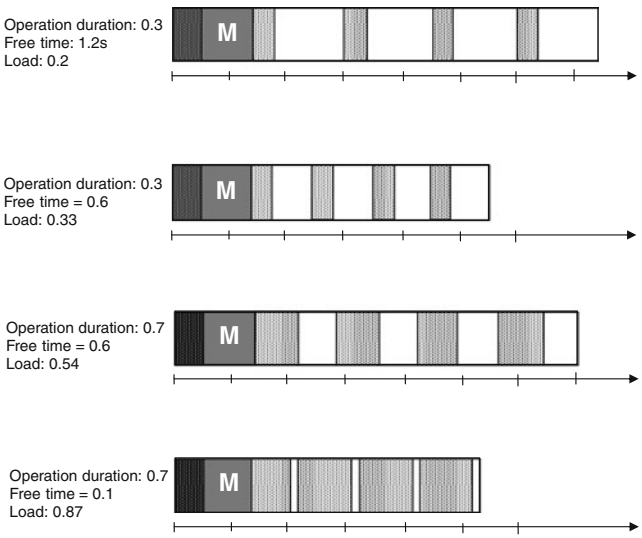

Fig. 2 Schematic illustration of four levels of cognitive load. The initial grey rectangle (labeled M) represents the presentation time of a memory item (1.5 s), of which the initial part (shaded rectangle, 0.5 s) is assumed in our model to be spent on encoding the item, the remainder being spent on refreshing. The following burst of four processing operations is subdivided into periods during which the operations are carried out (operation duration, *shaded rectangles*) and periods spent on refreshing (*white rectangles*)

se, however, is not critical. Rather, difficult tasks capture attention for longer time during each processing step, and if processing pace is reduced accordingly, memory performance can be higher with a difficult than with an easy processing task (Barrouillet et al., 2004; Barrouillet et al., 2007). (4) When processing difficulty and rate are held constant, the number of processing steps following each memory item has been reported to have no effect on memory accuracy (Barrouillet et al., 2004). This does not always hold, however, a point to which we will return in the last section of this article.

The TBRS explains the first finding by the assumption of decay during the processing periods that cannot be fully compensated by rehearsal. The remaining three findings follow directly from the cognitive load equation: Thus, as pace increases, t is reduced, implying higher cognitive load and thus a *decrease* in complex span performance. Likewise, as the processing task becomes more difficult, t_a increases, implying higher cognitive load. Finally, when attentional capture and pace are held constant (i.e., t_a/t is constant), the number of operations plays no role for cognitive load and hence should not affect memory. Across several experiments varying processing rate, processing difficulty, and the number of processing steps, cognitive load has been shown to be an excellent predictor of people's span (Barrouillet et al., 2004; Barrouillet et al., 2007; Vergauwe, Barrouillet, & Camos, 2009).

Attention-based refreshing in the TBRS must be distinguished from rehearsal as conceptualized in other theories of working memory, in particular Baddeley's (1986) concept of articulatory rehearsal in the context of the phonological loop model. Articulatory rehearsal is a domain-specific mechanism for recreating phonological memory traces by reproducing them through covert speech. Refreshing, in contrast, is a domain-general mechanism of reviving memory traces by attending to them (cf. Raye, Johnson, Mitchell, Greene, & Johnson, 2007); it does not involve articulation.

A study by Hudjetz and Oberauer (2007) provides direct evidence that the beneficial mechanism described as refreshing in the TBRS must be different from articulatory rehearsal. Hudjetz and Oberauer used a version of the reading span paradigm in which people read several sentences aloud and tried to remember the last word of each sentence. The pace by which sentence segments were presented was varied, and people's span was larger with the slower pace, as predicted by TBRS. Critically, this benefit of slower pace was obtained regardless of whether participants were allowed to read at their own rhythm—with pauses in between articulations—or had to read continuously to the pace of a metronome, even though continuous reading made it much harder to squeeze in articulation of additional material. The advantage of the slower processing pace thus does not hinge on the opportunity for articulatory rehearsal. Whatever the nature of the beneficial mechanism that improves memory at a slower processing pace, it must be able to operate concurrently with overt articulation of unrelated material.

Further evidence on the nature of refreshing in TBRS comes from a study by Barrouillet et al. (2007). They showed that memory span declines linearly with increasing cognitive load when the processing period consisted of a series of choice reaction time trials, but remained constant (and high) when it consisted of simple reaction time trials. Research in the dual-task literature using the psychological refractory period (PRP) paradigm has revealed that people have difficulties performing two unrelated speeded choice tasks at the same time, whereas simple reaction time tasks produce essentially no dual-task costs (Pashler, 1994). These findings point to a processing bottleneck for response selection, that is, the selection of one out of several responses, as required in choice reaction time tasks but not simple reaction time tasks. The findings of Barrouillet et al. (2007) therefore support their assumption that the bottleneck that is presumed to govern processing and refreshing in the complex span paradigm is the same as the response-selection bottleneck identified in dual-task studies with speeded choice tasks.

For a verbal theory, the TBRS is impressively clear and easy to intuit. Nevertheless, it leaves unspecified a number of details that have important implications for what the theory actually predicts—for example, how exactly does refreshing proceed? In what order are items refreshed? And at what rate? Even seemingly simple verbal models require numerous decisions to be made when instantiated computationally. For example, Lewandowsky and Farrell (in press, Chapter 2) showed that the phonological loop model of Baddeley (1986) could be instantiated in at least 144 different ways—thus, far from being "a" model, verbally stated theories typically are compatible with a whole family of instantiations, and detailed decisions must be made when the verbal theory is translated into a computational model. Because some of those decisions can have substantive consequences, we note them thoroughly while we present our instantiation.

TBRS*: a computational model of time-based resource sharing

The TBRS has so far only been applied to the complex span paradigm, so we focus on this paradigm at the outset. The methodological antecedent of the complex span task, immediate serial recall, has been the target of several computational models (Brown, Neath, & Chater, 2007; Brown, Preece, & Hulme, 2000; Burgess & Hitch, 1999,

2006; Farrell & Lewandowsky, 2002; Henson, 1998b; Page & Norris, 1998). We will draw on these existing theoretical achievements to guide and inform our modeling.

Our modeling strategy was to build a generic model of serial recall, using mechanisms from existing models of that task, and add to it mechanisms specific to the TBRS, namely time-based decay and refreshing. We aimed for parsimony during model construction to ensure that the model's behavior is governed by the core theoretical assumptions rather than any peculiarities of the implementation. Nevertheless, even a simple computational implementation of memory for serial order requires a number of decisions, and adding an attentional mechanism that rapidly switches between processing and refreshing, as assumed in the TBRS, brings with it a number of further modeling decisions. The most important modeling decisions are summarized in Table 1.

A first pair of decisions concerned the representation of items and their order. On pragmatic grounds, we opted for localist rather than distributed representations of items; that is, each item was represented by a single unique unit in a neural network, rather than as a pattern of activation across units. This simple representation is sufficient because the TBRS in its original formulation (Barrouillet et al., 2004) makes no assumptions about the internal structure of items or the similarity between them. We use 81 units to represent 81 different items, each item being represented by a unique active unit with all other units off.[1]

Regarding representation of order, we considered two options, chosen on the basis of their proven prowess in modeling serial recall (Farrell & Lewandowsky, 2004). One is coding of order by a primacy gradient (Grossberg & Pearson, 2008; Grossberg & Stone, 1986; Page & Norris, 1998), and the other is coding order by associating each item to a positional marker (G. D. A. Brown et al., 2000; Burgess & Hitch, 1999; Henson, 1998b). We investigated both options, but were successful only with the latter. Our failure to implement the TBRS using a primacy gradient to represent order illuminates principled limitations of primacy-gradient models, which we explain in Appendix A. We therefore opted for a model that uses position coding to represent order. This approach proved successful, and we next present this model, TBRS*.

Model architecture

The architecture of TBRS* is a two-layer connectionist network with one layer dedicated to the representations of positions and the other to the representation of the items. The two layers are fully interconnected, and their weights initialized at zero. To facilitate exposition of the theory, Table 2 lists the principal symbols used in our formal description of TBRS* and briefly outlines their function. Table entries are in the order in which symbols are introduced in what follows.

Position coding means that each item is associated to a "position marker." Position markers are pre-defined representations of serial position (e.g., Burgess & Hitch, 1999). Typically, neighboring position markers are assumed to be similar to each other. We use distributed representations of positions among which similarity (i.e., the degree of overlap of their activation patterns) decreases with distance. Specifically, we generate a random pattern for the first position, and derive each next position by changing a random subset of the features of the preceding pattern. In this way, the similarity between positions falls off by an exponential gradient, governed by a free parameter *P,* the proportion of units maintained from each position to the next.

Timing of cognitive processes

The TBRS model places great emphasis on the time-bound nature of all cognitive processes. To reflect this emphasis, we implemented every process (i.e., encoding, recall, refreshing, and distractor processing) as an exponential growth over time of some latent variable (e.g., learning strength, or evidence for a response), expressed generically as

$$x = 1 - \exp(-rt). \tag{1}$$

Here, x is the latent variable, r is the rate of processing (clipped at a minimum of 0.1 to avoid negative values), and t is the time spent on processing. The process finishes when the latent variable has reached a criterion τ. This basic model is borrowed from accumulator models of response time (S. Brown & Heathcote, 2005; Ratcliff & Smith, 2004; Usher & McClelland, 2001).

We do not simulate the accumulation process step by step. Rather, we use Eq. 1 and variants thereof to compute the duration of each processing step by solving for t. We treat t as a random variable and thereby simulate a distribution of process durations. This is accomplished by drawing processing rates r from a Gaussian distribution with mean R and standard deviation s [$r \sim N(R, s)$; we use capital letters for means and corresponding lower case letters for random variates]. In some applications of Eq. 1 we start from the mean processing rate R as a free parameter. In other applications we start from an estimate of the mean process duration T and work backwards to calculate R from it, then forward again to calculate

[1] The number 81 is arbitrary and makes little difference as long as there is a sufficiently large number of items that are not included in a given memory list, thus enabling extra-list intrusion errors (i.e., recall of items that were not on the list).

Table 1 Decisions for implementing TBRS as a computational model

Feature	Options	Option chosen
Representation of items	Localist/distributed	Localist
Representation of order	Primacy gradient/position coding	Position coding
Schema of overlap of position markers	Moving windows/exponential decline	Exponential decline
Encoding dynamics	Instantaneous/time-dependent	Time-dependent
Encoding strength	Bound by asymptote/unbound	Bound by asymptote
Decay function	Linear/exponential/...	Exponential
Use of item presentation time	Entirely devoted to encoding/partially used for refreshing	Partially used for refreshing
Mechanism of refreshing	Boost whole weight matrix/boost section of weight matrix associated to one position marker/retrieve and re-encode item/...	retrieve and re-encode item
Schedule of refreshing	Refresh last item encoded/refresh randomly selected item/ cumulative in forward order	Cumulative in forward order (resume with 1st item after every interruption)
Response suppression after overt recall	None/removing item from weight matrix/suppress item in item layer	Remove item from weight matrix
Response suppression after retrieval for refreshing	None/removing item from weight matrix/suppress item in item layer	None

Table 2 Symbols in the formal notation of TBRS*, their meaning, and their function

Symbol	Meaning	Function
P	Position marker overlap	Free parameter; proportion of units maintained from each position to the next
η	Learning strength	Determined by time available for encoding or refreshing (Eq. 3)
L	Weight asymptote	Asymptote for weights connecting context to items, set to 1/9 (there were 9 units in each context marker) throughout
r	Momentary memory processing rate	Determines unfolding of learning strength (η) over time (Eqs. 3 and 3a) for initial encoding and refreshing, and speed of recall; r is distributed normally with mean R and standard deviation s, and minimum .1
τ_E	Criterion for encoding strength	Free parameter, determines learning strength during encoding via Eq. 3a
R	Mean memory processing rate	Free parameter governing mean speed of encoding, refreshing, and recall
s	Standard deviation of processing rates	Free parameter governing variability in the duration of all processes (i.e., encoding, refreshing, distractor processing, and recall)
t_e	Encoding time for an item	Obtained by Eq. 3a based on R, s, and τ_E
θ	Retrieval threshold	Free parameter that determines minimum level of activation required for an overt retrieval
σ	Standard deviation of Gaussian noise added to item activations at retrieval.	Free parameter that determines likelihood of extra-list intrusions
η_{al}	Strength of anti-learning	Determines strength of response suppression; set to L
D	Decay rate	Free parameter that determines rate of decay of weights towards zero
τ_R	Criterion for refreshing strength	Determines learning strength during attentional refreshing, derived from T_r and R via Eq. 7
T_r	Mean time taken to refresh an item	Free parameter that determines the mean duration of each refreshing step
t_r	Time for an individual refreshing event	Obtained by Eq. 3a based on R, s, and τ_R
r_{op}	Momentary distractor processing rate	Determines rate of evidence accumulation and hence time of attentional capture of a distractor processing step (Eq. 8); r_{op} is distributed normally with mean R_{op} and standard deviation s
R_{op}	Mean distractor processing rate	Computed through Eq. 8a from estimates of mean processing duration, T_a, and τ_{op}
τ_{op}	Response criterion for processing	Processing finishes when evidence accumulator reaches this value. Set to τ_E
T_a	Mean duration of attentional capture by distractor processing steps	Estimated from measured response times or set according to theoretical assumptions
t_a	Momentary duration of attentional capture by a distractor	Computed by solving Eq. 8

individual processing durations t, as described in more detail below in the context of each particular process.

Encoding

When an item is presented, its representation—a single unit representing that item—is activated in the item layer, and at the same time, the representation of the current list position—that is, the current positional marker—is activated in the position layer. The item is associated to the position by Hebbian learning:

$$\Delta w_{ij} = (L - w_{ij})\eta a_i a_j, \tag{2}$$

where w_{ij} is the connection weight between unit i in the position layer and unit j in the item layer, a_i is the activation of unit i, a_j is the activation of unit j, and η represents the learning strength (determined by Eq. 3 below). The product of the activations in the two layers is multiplied by $(L - w_{ij})$ so that over multiple learning events the weight grows towards an asymptote L. For all our simulations, L was set to 1 divided by 9, the number of active units in each position marker.[2] Activations a_i and a_j were always 0 or 1 during encoding, for units that were off or on, respectively. Because we use localist representations of items and distributed representations of positions, each item-position association consists of strengthening the weights between several position-layer units and a single item-layer unit.

We model the duration of each encoding event by using a variant of Eq. 1 in which learning strength η figures as the latent variable that increases with time:

$$\eta = 1 - \exp(-rt_e). \tag{3}$$

Here, r is the rate of memory encoding, and t_e is the time spent on encoding. By Eq. 3 the learning strength has an asymptote of 1, so that in combination with Eq. 2, the strength of connection weights grows towards an asymptote L.

We assume that each encoding event proceeds until the growth of learning strength η has reached a criterion level τ_E. As stated above, encoding duration t_e is a random variable that varies from one encoding event to the next. This variability is implemented by drawing the encoding rate r from a Gaussian distribution with mean R and standard deviation s; R and s are free parameters of the model. We obtain the duration of each encoding event by setting $\eta = \tau_E$ in Eq. 3, inserting the value of r drawn from the Gaussian distribution, and solving for t_e:

$$t_e = -\frac{\log(1 - \tau_E)}{r} \tag{3a}$$

As long as t_e, computed by Eq. 3a, does not exceed the presentation time for an item, the growth of η will have reached the criterion τ_E when encoding stops, so that we can set $\eta = \tau_E$ in Eq. 2 to compute the change of connection weights. In those cases where t_e exceeds the presentation time, it is clipped to the presentation time, and η is computed from Eq. 3. Figure 3 presents an illustration of the relation between t_e, R, and τ_E.

To summarize, in the simulations we use R, s, and τ_E as free parameters. For each encoding event we draw a random value r ($r \sim N(R, s)$ and $r \geq 0.1$), compute t_e from it using Eq. 3a, and insert τ_E for η in Eq. 2 to determine the learning strength (except for $t_e >$ presentation time, when η is computed from Eq. 3). We assume that the system tries to encode the items strongly, so we set τ_E to 0.95 for all simulations. Jolicoeur and Dell'Acqua (1998) showed that consolidation of information in short-term memory is completed after about 0.5 s, which implies that the limiting value of τ_E, .95, should be achieved after 0.5 s. It turns out that this corresponds to a mean rate of $R = 6$. Thus, the value of this crucial parameter was set on the basis of independent

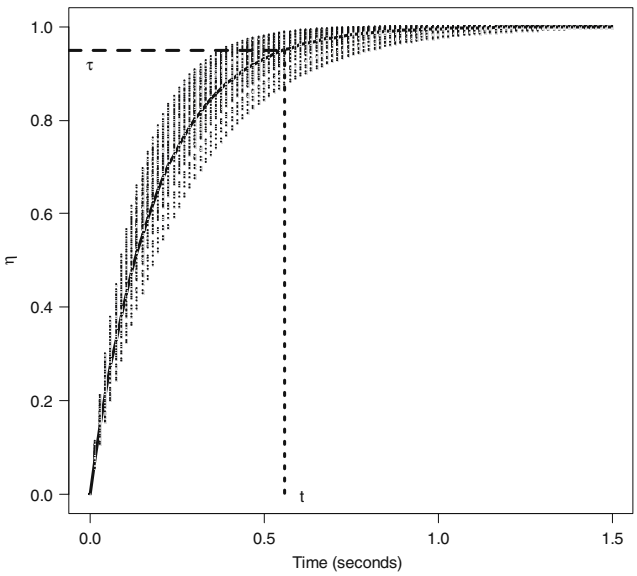

Fig. 3 The exponential growth of learning strength η as a function of time. The *hatched area* represents the distribution of 100 curves generated from 100 samples of rate r drawn from a normal distribution with mean $R = 6$ and standard deviation $s = 1$, representing 100 different occasions. The *continuous line* represents one such occasion selected at random. The duration of this event, t, is determined by the time it takes the curve to reach the criterion, τ. This dynamic applies equally to encoding and recall (with τ_E as the criterion τ), refreshing (with τ_R as criterion), and distractor processing (with τ_E as criterion and R_{op} instead of R)

[2] Defining the learning asymptote as the inverse of the number of active units in the retrieval cue (i.e., the positional marker) has the advantage that activation strength of any unit in the item layer when retrieved through the weight matrix has a maximum of 1, reached when the item unit is associated to all units of the cue with the maximum weight.

empirical findings. We set the standard deviation of encoding rates (s) to 1, which results in a standard deviation of encoding times of approximately 0.095 s.

In many experiments items are presented for much longer than 0.5 s. Barrouillet et al. (2004), for instance, used 1.5 s per item. We assume that when encoding of an item is completed, the cognitive system uses the remaining presentation time for refreshing in the same way as it uses free time periods in between processing operations (more on refreshing below).

Recall

When it comes to recalling the items, the position markers are again activated in the position layer one by one in forward order. Each position marker forwards activation through the weight matrix to the item layer, thus generating a profile of activation across the item units:

$$a_j = \sum_{i=1}^{n} a_i w_{ij}. \quad (4)$$

Here, a_j is the activation of a unit in the item layer, a_i the activation of a unit in the position layer, w_{ij} the weight connecting them, and n the number of units in the position layer. Typically, the unit representing the item that was actually presented in a given position receives the highest activation, and units of other items that were associated to neighboring positions also receive some activation. The unit with the highest activation is selected and recalled. Occasionally, the unit representing a different item receives the highest activation, resulting in an error.

Item errors

Unless further assumptions are made, the recall process just described cannot generate item errors, such as omissions (a "pass" response) or intrusions (recall of a non-presented item). This limits the applicability of the model to experiments that minimize item errors (i.e., using a small closed pool of items and forcing participants to give a response at every position). We therefore augmented the basic item-selection process to allow for the occurrence of item errors. In the first step, we introduced a threshold for retrieval as a new parameter, θ. If after cueing no item's activation exceeds the threshold, an omission error occurs. This model produced a substantial number of omission errors but still no extralist intrusions. Extralist intrusions can occur only if an item not on the list accrues a positive activation value at retrieval, so that it can surpass all list items' activation. The model architecture with 81 units in the item layer affords localist representations of 81 different items, thus providing room for extralist intrusions. However, the units representing extralist items are not associated to any position representation, and thus receive no activation through the weight matrix. One straightforward way to activate extralist items at retrieval is by adding noise to all units of the item layer at every retrieval attempt. This required a further parameter, σ, which specified the standard deviation of this Gaussian noise. Except where noted otherwise, we set $\theta = 0.05$ and $\sigma = 0.02$ in the simulations reported in this article.

Response suppression

After overt output, the recalled item is suppressed, which minimizes its accessibility for the remainder of the recall episode. There is widespread agreement that some form of response suppression is needed to accommodate a variety of findings in serial recall (e.g., Farrell & Lewandowsky, 2002; Page & Norris, 1998). For example, erroneous repetitions of items during recall are exceedingly rare, and even when list items are repeated, people are often unable to report both occurrences of the repeated item (the Ranschburg effect, see, e.g., Henson, 1998a). We instantiated response suppression by Hebbian anti-learning (Farrell & Lewandowsky, 2002): Whereas in Hebbian learning, the product of activations in position layer and item layer is added to the weight matrix, in anti-learning it is subtracted:

$$\Delta w_{ij} = -\eta_{al} a_i a_j. \quad (5)$$

For calculating the weight changes for response suppression, only the activation of the item selected for output is maintained in the item layer; all other activations are set to zero to avoid suppressing items that were not recalled. The learning strength for anti-learning, η_{al}, was set equal to the asymptote of learning, L, to make sure that anti-learning removed an item at least as strongly as it has been encoded.

Recall takes time. Data from serial recall experiments point to latencies between 0.5 and 1 s per item for spoken recall (somewhat longer for typed recall), with the exception of the first item, which takes considerably longer to recall (Dosher & Ma, 1998; Farrell & Lewandowsky, 2004; Maybery, Parmentier, & Jones, 2002). Recall times in complex span tasks have rarely been measured, and they seem to depend on the specific version of complex span, with longer recall times for reading span or listening span than for counting span (Cowan et al., 2003). Recall times are potentially important in the TBRS because during recall memory continues to decay. For our simulations we set mean recall duration per item to a lower bound estimate of 0.5 s, thus potentially underestimating the amount of decay during recall. The recall times were simulated as random variables in the same way, and using the same parameter values, as the encoding times.

Decay and refreshing

Because memory for a list is represented in the weight matrix, we assume exponential decay with rate D to affect the weights w_{ij},

$$\frac{dw_{ij}}{dt} = -Dw_{ij}. \quad (6)$$

Decay is applied to the whole weight matrix during every time interval of the simulation. For most of our simulations we set the decay rate $D = 0.5$ (all parameter values used in all simulations are summarized in Table 3) because this value enabled a satisfactory reproduction of the cognitive-load effect, as shown below.

Refreshing is modeled as retrieval followed by (re-) encoding of the retrieved item. In every period of free time, the attentional mechanism attempts to refresh as many items as possible one by one. Each refreshing step starts with retrieving an item exactly as for recall, with the exception that no response suppression through anti-learning is applied. Response suppression during refreshing would undermine the purpose of refreshing because it removes the very memory trace that should be strengthened. The item that is retrieved at each refreshing step is strengthened by associating it to the current position marker exactly as during encoding of new items, the only difference being that the system takes less time for refreshing each item than for its initial encoding. The duration of each refreshing step is governed by the same process that governs duration of initial encoding, expressed by Eq. 3a. We used the same rate, R, for refreshing and for encoding, because we assume that refreshing strengthens memory at the same rate as initial encoding. However, as we will demonstrate below, refreshing must be assumed to proceed very rapidly from one item to the next. For that reason we did not use the criterion of initial encoding, τ_E, but introduced a separate criterion τ_R ($\tau_R << \tau_E$) to govern the duration of a refreshing step, replacing τ_E in Eq. 3a. Thus, refreshing proceeds at the same rate as initial encoding but finishes much earlier, after reaching a much lower criterion. For the simulations we found it convenient not to treat the criterion τ_R as a free parameter, but rather to use the mean duration of a refreshing step, T_r , as a free parameter and compute τ_R from it and the mean processing rate R:

$$\tau_R = 1 - \exp(-RT_r). \quad (7)$$

Equation 7 is another variant of our basic model of latencies as expressed in Eq. 1, this time written in terms of variable means (i.e., inserting R for r, T_r for t, and τ_R, the value of learning strength reached when the refreshing step finishes, for x). This form of the equation is convenient because it enables the computation of τ_R from T_r. We set T_r to 0.08 s per item, implying a value of $\tau_R = 0.38$ for refreshing. Thus, each refreshing step increases the strength of the refreshed item with a learning strength of 0.38, considerably less than the learning strength at initial encoding, which typically equals the criterion $\tau_E = 0.95$.

Refreshing within a burst proceeds in a cumulative fashion, that is, it starts from the first list item and proceeds in forward order until the end of the list as encoded so far, then starting over at the beginning. Whenever refreshing is interrupted by a processing operation or a new item, it starts again with the first list item when resumed.

Processing

In between encoding of items, a series of processing operations must be carried out (e.g., simple arithmetic computations or two-alternative forced-choice tasks). We determine the processing duration t_a (i.e., the time for which a processing step captures attention) by another variant of our basic accumulator model, assuming that an

Table 3 Parameter values for simulations

Parameter	Sim. 1	Sim. 2	Sim. 3 (TBRS^0)	Sim. 4 (8y/14y)	Sim. 5	Sim. 6	Sim. 7
Position marker overlap (P)	0.3	0.3	0.3	0.3	0.3	0.3	0.3
Criterion for encoding (τ_E)	0.95	0.95	0.95	0.5/0.55	0.95	0.95	0.95
Processing rate (R)	6	5	6	6	6	6	6
SD of processing rate (s)	1	1	1	1	1	1	1
Decay rate (D)	0.5	0.5	0.5	0.5/0.4	0.35	0.31	0.5
Refreshing duration (T_r, in ms)	80	80	80	80	80	80	80
Threshold for retrieval (θ)	0.1	0.1	0	0.1	0.1	0.05	0.1
Noise (σ)	0.02	0.02	0	0.02	0.02	0.02	0.05

Note: Sim = simulation, 8y = 8-year-old children, 14y = 14-year-old children. All simulations other than Simulation 3 involve TBRS*

evidence accumulator e for the chosen response grows over time, driven by the processing rate r_{op}:

$$e = 1 - \exp(-r_{op}t_a). \tag{8}$$

The operation is completed when the accumulation value e reaches a criterion τ_{op}, which for simplicity we set to 0.95, the same value as τ_E. The rate r_{op} is drawn from a Gaussian distribution with mean R_{op} and standard deviation s. Experiments and experimental conditions differ in the difficulty of the operations to be carried out, and we model the difficulty of operations through variation of R_{op}.

To calculate the necessary variables for the simulations, we start from an estimate of mean processing duration T_a for a given experimental condition. Where available, we used measured response times (T_a) as an estimate of the mean duration of attentional capture by an operation, following Barrouillet et al. (2007) and Portrat, Camos, & Barrouillet, (2008). However, we do not use the mean duration as the duration of each individual processing step, because we want to simulate a distribution of durations of attentional capture, analogous to the distribution of durations of other processes (i.e., encoding, refreshing, recall) in TBRS*. Therefore, the next step is to calculate the mean processing rate R_{op}, using Eq. 8. Specifically, we set the accumulated evidence e in Eq. 8 to the value it has reached when processing finishes, τ_{op} , solve for r_{op}, and replace r_{op} by its mean, R_{op}.

$$R_{op} = -\frac{\log(1-\tau_{op})}{T_a}. \tag{8a}$$

Finally, the duration of each individual operation is computed from Eq. 8: We set e to τ_{op}, the value that the accumulator has to reach for an operation to finish. We draw a momentary processing rate r_{op} from a normal distribution [$r_{op} \sim N(R_{op}, s)$]. With these values we solve Eq. 8 for t_a to obtain the time at which the evidence accumulator reaches the criterion and the operation finishes.

Process scheduling

The complex span paradigm requires many modeling decisions about which processes occur when. Among the most important initial decisions is a consideration of the nature of task switching in a complex span task. Any complex span task involves at least two tasks, namely encoding of the memoranda and processing of the distractors. In the TBRS, there are possibly additional task switches between encoding of an item and its refreshing, and also between processing and refreshing. How are those switches best modeled in light of the extensive task-switching literature? There is pervasive agreement that any switch between tasks involves a cost, usually measured by the additional time it takes to complete a new task as opposed to another round of the same one (Monsell, 2003). Recently, Liefooghe, Barrouillet, Vandierendonck, and Camos (2008) provided evidence that the attentional bottleneck is occupied during that switch time; decisions about how task switching is to be modeled are therefore crucial to our instantiation of the TBRS.

We assume from here on that switching between encoding, processing, and refreshing occupy the attentional bottleneck for some non-negligible duration. Unless specified otherwise, in all simulations that follow we subsume this switch cost within the duration of distractor processing, which thus reflects a composite of (1) task switching *to* the processing task from encoding or refreshing, (2) switching *away from* the processing task to refreshing or encoding the next item, and (3) doing the processing task proper, that is, selection of a response to the current distractor. A further decomposition into those separate components is typically unnecessary; an exception will be Simulation 7.

We next make explicit the processes that are engaged at each point during a trial of the type of complex span task used by Barrouillet et al. (2004, 2007). The model readily generalizes to other tasks that interleave memorization and processing; generalization beyond that family of tasks to other working-memory paradigms such as running memory (Bunting, Cowan, & Saults, 2006; Pollack, Johnson, & Knaff, 1959) or memory updating (Ecker, Lewandowsky, Oberauer, & Chee, 2010; Kessler & Meiran, 2006) require more substantial modifications and therefore more theoretical decisions on what is assumed to happen, and when, and we therefore do not apply the model to those tasks here.

A temporal trace of activity

The events during a complex span trial can be illustrated by the trace of an example trial. Figure 4 shows two such traces of memory strength over time for each item in a seven-item list. Memory strength is defined as the strength of association of each item to the representation of its list position, which in turn determines the activation of that item in the item layer when cued by its list position (as per Eq. 4). Each panel of Fig. 4 shows seven items being encoded, each followed by a burst of processing, and some items being successfully recalled (recall begins at the point where activations fall below baseline owing to response suppression; i.e., at $t = 37$ and $t = 48$ in the top and bottom panels, respectively.)

Each trial begins with the encoding of the first memory item by associating it to the first position marker. Encoding continues until the encoding criterion (τ_E in Eq. 3a) has been reached. The remainder of the presentation time is devoted to refreshing. After encoding of the first item, refreshing can only apply to that item, so refreshing

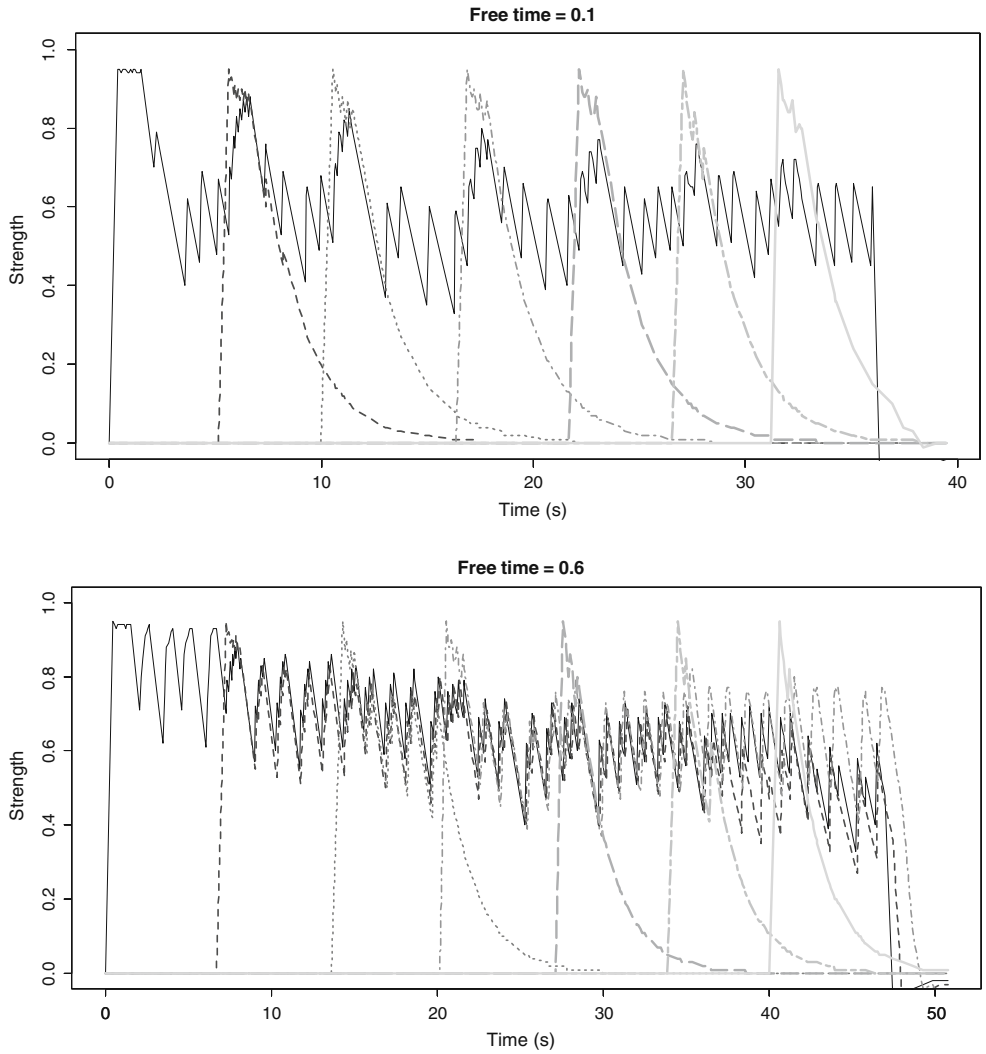

Fig. 4 Traces of memory strength over time for a seven-item list with four operations of 0.7 s each following each list, and free-time periods of 0.1 s (*top panel*) or 0.6 s (*bottom panel*) following each operation. Memory strength is defined as the strength of association between the position marker at position p and the list item presented at position p; different line types represent strengths of different items across the seven positions. Each item rapidly gains strength during encoding (within about 0.5 s of its presentation time). The remaining presentation time is devoted to refreshing of all items encoded so far. During a processing operation, all memory items decay, noticeable by the steep decline of all strengths. During the following period of free time, memory items are refreshed. Refreshing rapidly cycles through all currently encoded items, and this is reflected in the tiny peaks and troughs of each item during the periods of refreshing: Each item peaks when it is refreshed and drops again while other items are refreshed. Occasionally, an item fails to be retrieved for refreshing and drops out; the gradual decelerated decline illustrates the effect of decay when unbridled by refreshing. After the final burst of processing operations, items are recalled and then suppressed (which can reduce their strength to be below baseline; thus the beginning of recall can be identified by the "below-zero" dips of activation). During recall of each item, the remaining items continue to decay

effectively means continued encoding of the first item, which is of little consequence because the item has reached nearly asymptotic memory strength already. During the presentation time of later memory items, refreshing in the remaining presentation time includes earlier items and thereby contributes to counteracting decay of these earlier items.

After presentation of each memory item, a varying number of processing operations are carried out (e.g., a series of simple arithmetic operations). The duration of each processing operation is determined by the mean processing rate, which is the same for each operation in a trial, together with the random variation on processing rate. The attentional bottleneck is occupied for the duration of each operation. In the free time between processing operations, which arises if the actual processing time falls short of the total allotted time, attention is devoted to refreshing memory items. The time schedule of one processing burst is illustrated in Fig. 2 for different levels of cognitive load.

The strength traces in Fig. 4 show how during each operation, memory strength decays, and during each period of free time, strength partially recovers through refreshing. Occasionally, however, an item is lost to irrevocable decline. This happens when during a refreshing attempt in position p the item originally presented in position p fails to be retrieved because another item q has achieved stronger activation during retrieval. As a consequence, refreshing strengthens item q in position p, thus giving it a further advantage over the correct item p. When that happens, it becomes extremely difficult to recover the lost item: Every time the representation of position p is re-instated as a retrieval cue, item q will most likely come out with higher activation than item p.

Examination of refreshing schedules

We instantiated a refreshing schedule that started from the first list item and proceeded in forward order, resetting to the first item whenever refreshing was interrupted. The decision to use this refreshing schedule was not arbitrary but the selection was based on consideration of several alternative refreshing schedules; because refreshing is pivotal in the TBRS, some of those alternatives—including those that failed—deserve to be highlighted. In a variant of the scheme just described, cumulative refreshing continues with the next list item after being interrupted by an operation and starts over with the first item only when a new item is encoded. This variant generated very similar data patterns to the one we adopted, but with slightly reduced overall performance and more pronounced non-monotonicity in the span-over-load curve (discussed below). Other refreshing schemes resulted in more drastic changes. For instance, it could be assumed that refreshing during each processing burst focuses on the last item encoded. This would leave the list-initial item to decay once later items have been encoded, thus resulting in a strong recency effect but no primacy effect, contrary to the data (shown below). Another scheme would be to pick items for refreshing at random with equal probability for all items encoded so far. This scheme would refresh recent items as much as earlier items, and because earlier items had more time to decay, their memory strength would be weaker than that of more recent items, resulting again in a pronounced recency effect with little or no primacy, contrary to the data. Another option, which appears plausible at first glance, is to refresh only those items that were below maximal activation, skipping over those whose activation had already reached asymptote. However, this selective-refreshing scheme would still require a stepping through the list to retrieve the items in order before their strength can be ascertained. Thus, the processing steps of the selective-refreshing approach are identical to the one we implemented, and its outcome is identical as well, because items whose strength is close to asymptote do not gain strength from refreshing whether they are skipped or not.

Our exploration of various refreshing schedules confirms an earlier analysis (Oberauer & Lewandowsky, 2008) that the exact effects of rehearsal are tied to the particular scheme being implemented, and that a blanket appeal to rehearsal or refreshing as a panacea for memory restoration is inadvisable.

One consequence of cumulative refreshing is that the distribution of refreshing time over memory items across a trial is very uneven. Before the second item is presented, all refreshing is concentrated on the first item. After presentation of the second item, refreshing time is divided roughly equally between the first two items; after presentation of the third item, refreshing time is divided roughly equally between the first three, and so on. Still the first item receives more refreshing than later ones because often a cycle of refreshing steps starts with the first item but is interrupted before reaching the current end of the list. This uneven distribution of refreshing is the reason why we imposed an asymptote to the strength of associations, and thus on the strength of memory representations in the model. Without that asymptote, refreshing could boost an item's strength without bounds, and this would imply that, during the processing burst in between presentation of the first and the second item, the first item is strengthened to a degree far beyond that reached by its initial encoding (especially when cognitive load is low). As a consequence, the strength of the first item becomes so overbearing that it intrudes in retrieval of the second item (due to overlap of the second with the first position marker). In overt recall, intrusions from preceding items are typically avoided by response suppression, but for reasons discussed above (and in Appendix A), response suppression has to be switched off during refreshing. Therefore, successful refreshing of a list of items requires that the memory strengths of neighboring items are not very different, which implies the need for an asymptote in memory strength.

Recall schedule

After the last processing burst, recall commences immediately. Items are recalled in forward order, each with the same mean duration. The assumption of equal mean recall times is a simplification, because recall times vary between list positions. In particular, latency of recall for the first list item is much longer than that of later list items (Farrell & Lewandowsky, 2004). We investigated model versions that spend one second on refreshing the whole list before commencing recall to account for the much longer latency of the first item, but that assumption had no noticeable impact on the model's behavior, so we dropped it for simplicity.

Each recalled item is suppressed; the time for suppression is included in the duration of recall. Suppression can be witnessed in Fig. 4 as the sharp drop of an item's strength slightly below baseline (beginning at $t = 37$ and $t = 48$ in the top and bottom panels, respectively).

Simulation of benchmark findings with complex span

In what follows we present several simulations to test the model. In the present section we investigate how well TBRS* predicts benchmark findings that have been cited in support of the TBRS. Simulation 1 applies the model to a hypothetical experiment with the complex span paradigm combining the three experimental variables that, when manipulated separately in different experiments, created these benchmark findings. Simulation 2 applies the model to the experiment with the most extensive variation of cognitive load so far, Experiment 7 in Barrouillet et al. (2004). This is followed by a detailed analysis of the model's behavior in terms of serial-position curves and types of errors, including a downscaled version of the initial model without the additional apparatus for item errors ($TBRS^0$; Simulation 3). In the following section, we present applications of TBRS* to the development of working memory (Simulation 4), to the Brown-Peterson paradigm (Simulations 5 and 6), and to a series of experiments with the complex span task that challenge the notion of decay (Simulation 7). Table 3 presents a summary of the parameter values for all simulations.

Simulation 1: A comprehensive complex span experiment

As noted earlier, four benchmark findings provide the core empirical support for the TBRS: (1) Adding a processing demand to a serial-recall task, thus turning simple span into complex span, impairs memory. (2) Slowing the processing pace in a complex span task benefits memory. (3) More difficult and therefore longer processing steps result in worse memory. (4) As long as cognitive load is held constant, the number of operations does not affect memory.

Simulation 1 applies the model to a hypothetical experiment that manipulates the three variables responsible for these findings: Number of processing steps in each burst (with levels 0, 1, 4, and 8); task difficulty, operationalized as the duration of each processing step (parameter t_a in the cognitive-load equation, with three levels, easy: 0.3 s, medium: 0.5 s, and hard: 0.7 s); and the free time following each processing step, during which attention can be devoted to refreshing, with five levels: 0, 0.1, 0.6, 1.2, and 2.0 s. The cognitive load is the ratio of processing duration t_a to the processing time t available for each step, where t equals t_a plus the free time. Our experiment does not emulate the earlier experiments of Barrouillet and colleagues, in which processing pace was manipulated while holding the total available processing time constant (i.e., by varying the number of processing steps squeezed into that time) or while holding the number of processing steps constant (i.e., by varying the total time available for all processing steps of a burst). That approach necessarily confounds two of the three experimental variables. Our hypothetical experiment varies these three variables independently. One way to accomplish this in practice is to measure processing duration t_a online through people's response times in each processing step and add a fixed interval of free time after each response, as done by Portrat, Camos, and Barrouillet (2008). Alternatively, mean processing duration t_a for each condition can be measured offline, and the total time for each processing step is determined by adding the desired free time to that mean (Barrouillet, Gavens, Vergauwe, Gaillard, & Camos, 2009).

By crossing three levels of processing duration and five levels of free time, we obtained 15 levels of cognitive load, varying from 0.13 to 1.0 (disregarding the simple span condition with 0 processing steps, for which cognitive load is undefined). Table 4 summarizes the 15 levels of cognitive load together with the operation durations and free times that jointly constitute each level. The simulation involved 200 simulated subjects, each of which went through a span procedure for each condition; the span procedure was modeled after Barrouillet et al. (2004). For each condition there were three trials for each list length from 1 to 9. The subject received 1 credit for each list reproduced perfectly in order, and 0 for each erroneous list. Credits were averaged within list lengths and summed

Table 4 Levels of cognitive load for Simulation 1 and Appendix B

Load	Operation duration (s)	Free time (s)
0.13	0.3	2.0
0.20	0.3 / 0.5	1.2 / 2.0
0.26	0.7	2.0
0.29	0.5	1.2
0.33	0.3	0.6
0.37	0.7	1.2
0.45	0.5	0.6
0.54	0.7	0.6
0.75	0.3	0.1
0.83	0.5	0.1
0.88	0.7	0.1
1.0	0.3 / 0.5 / 0.7	0

Note: Combinations of operation duration and free time that result in the same cognitive load are listed in a single row.

across list lengths. On this scoring technique, the maximum span that could be achieved was 9.

Simulation 1 evaluated TBRS* on the comprehensive complex span experiment described above. The most important results are shown in Figs. 5 and 6. Figure 5 shows span as a function of cognitive load (top). The simple span condition (number of operations = 0) is plotted as a reference for all levels of cognitive load. The figure shows that span declines in a roughly linear fashion with increasing cognitive load. Only at the lowest level of load does complex span approach simple span.

Memory performance is also often expressed as proportion of items recalled in correct position (e.g., Vergauwe et al., 2009). Therefore, Fig. 5 (bottom) plots this alternative dependent variable as a function of cognitive load. Again, memory performance declines in an approximately linear fashion with increasing cognitive load. Thus, TBRS* generates the most important prediction derived from the verbal TBRS by its authors, the roughly linear decline of memory performance with increasing cognitive load. The simulation enables us to investigate the processes leading to this prediction in detail, and this is what we do next.

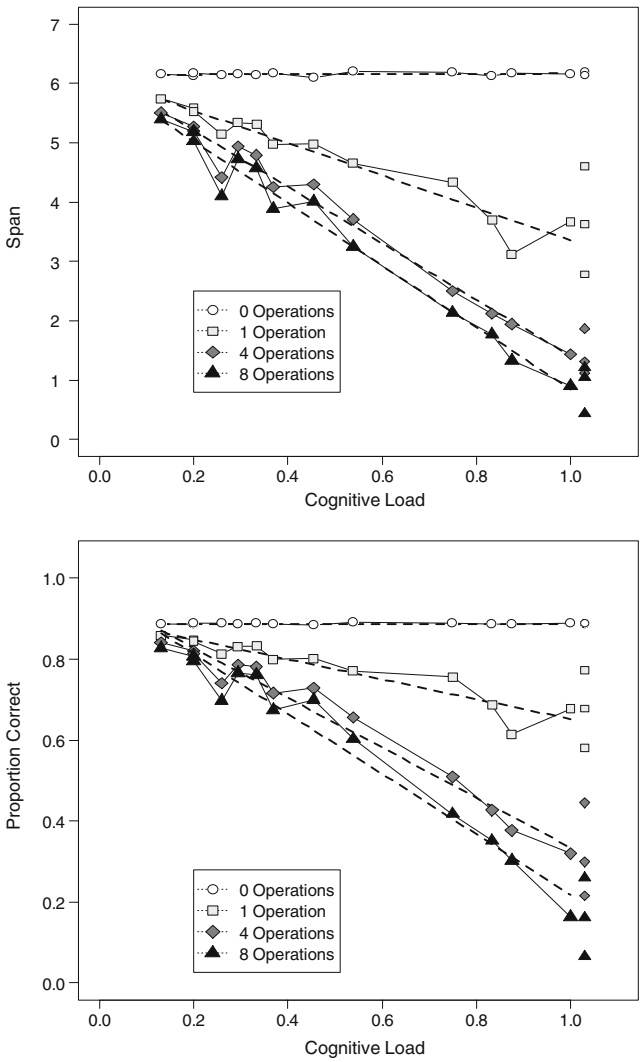

Fig. 5 Simulation 1: Span as a function of cognitive load (*top*) in TBRS*. Proportion correct across all list lengths as a function of cognitive load (*bottom*). The highest level of cognitive load is represented by three data points for each level of number of operations, because for all three operation durations, load = 1 when free time = 0

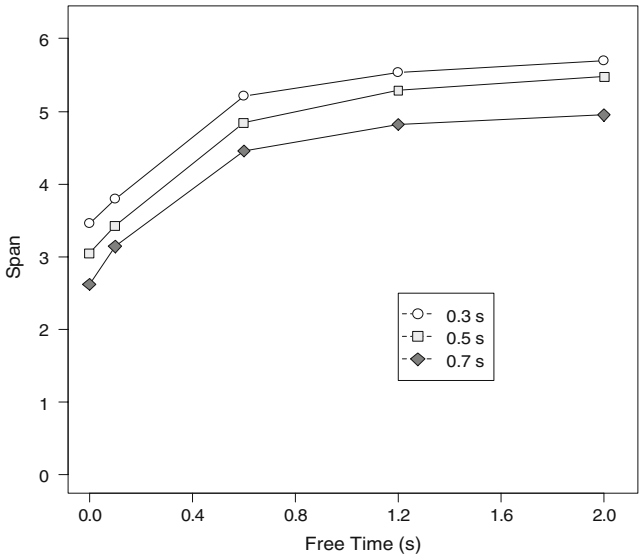

Fig. 6 Simulation 1: Span as a function of free time and operation duration in TBRS*. Data are from the condition with eight operations per burst

Three observations are worth noting in Fig. 5. First, the data points for simple span (at the top of each of the panels) reflect 15 identical replications that differ only with respect to randomization. The variation between those data points thus reflects the degree of random noise in the simulated spans. It is clear from the figure that random variability is quite small.

Second, the decline of memory performance with cognitive load is not entirely monotonic. The deviations from monotonicity arise from the fact that the effects of operation duration and of free time on memory are not linear, as shown in Fig. 6. In particular, increasing free time has diminishing returns, so that at a low level of cognitive load (towards the right in the figure), where free time is already relatively large, adding more free time leads to only small benefits for memory. At the same time, increasing operation duration still incurs a substantial decay cost even at high levels of free time—compare the lines for operation durations 0.7 s vs. 0.3 s. Thus, when a long free time is combined with a long operation duration, the span predicted by TBRS* is lower than the span predicted by

interpolating from a linear function of cognitive load. This explains the discontinuities in the span-over-load function: Those cognitive-load levels for which performance dips below what would be expected from a linear function of load are the ones with the longest level of free time (see Table 4).

The diminishing benefit of free time together with the continuing cost of longer operation durations implies that refreshing cannot fully compensate decay. This observation points to the limited power of refreshing and rehearsal that we already noted in a previous modeling study (Oberauer & Lewandowsky, 2008). Refreshing is not the inverse operation of decay. Rather, the effect of refreshing depends on the accuracy of retrieving each item in its correct position – to the degree that retrieval fails, refreshing will strengthen the wrong item-position association and thereby damage memory rather than improving it. Longer operation durations imply longer decay and thereby make it more likely that an item cannot be retrieved during the next attempt at refreshing it. If that happens, the item is irrevocably lost, and no addition of free time can resurrect it (see Fig. 4). For those items that still are retrieved correctly, increasing free time is beneficial only up to a point. As their strength approaches asymptote, further refreshing adds increasingly little strength, while still carrying the (albeit tiny) risk of retrieval failure. Taken together, increasing free time yields diminishing returns because it cannot help items that have already decayed beyond recovery, whereas those that can still be retrieved soon cease to need help.

The final observation to be made with respect to Fig. 5 is that memory performance declines with increasing number of operations. This effect is hardly noticeable at low levels of cognitive load but becomes substantial at higher levels of load. This is a point where the prediction of TBRS* deviates from the predictions that Barrouillet et al. (2004) derived intuitively from the verbal formulation of their theory. The intuitive prediction is that span is determined only by cognitive load, and as long as cognitive load is held constant, the number of operations does not matter. This intuition derives from the idea that at a given level of cognitive load, decay and refreshing will keep a balance that is stable over repeated decay-refreshing cycles, and therefore does not change with the number of operations. This intuition is approximately true at low levels of cognitive load, where refreshing nearly completely reverses the effect of decay, thus approximately restoring the state before decay. At higher levels of cognitive load, however, refreshing falls short of restoring the pre-decay state of memory. Rather, the net effect of decay during one operation and refreshing during one interval of free time after that operation is negative, and adding more operations therefore results in worse memory.

At first glance, this behavior of the theory stands in conflict with published data that show that adding more operations in between memory items does not affect performance, even at very high cognitive loads (Oberauer & Lewandowsky, 2008). We revisit the issue of the number of operations later while discussing some additional recent data (Simulation 7); for now, it suffices to note that with some auxiliary assumptions TBRS* can handle at least some aspects of these results, permitting us to set aside that discrepancy for the time being.

Figure 7 shows the dynamics of decay and refreshing close up. The line traces the strength of one item during a burst of eight operations, starting at 1 and decaying during operations, being refreshed in the free time in between. The free time is assumed to be shared equally between five items, so that only 1/5 of the nominal free time is dedicated to refreshing the plotted item. The four panels show four levels of cognitive load. At low load (top left panel), refreshing nearly entirely compensates decay, and the item's strength does not decline over successive processing operations. At higher load (other three panels), in contrast, refreshing fails to fully compensate for decay, and strength gradually declines over successive operations. Irrespective of the values of D and R, which differ between panels, the decline eventually levels off, reaching a steady state. Therefore, once a number of operations have been completed, adding more operations makes increasingly little difference for memory strength. The reason for this convergence to a steady state lies with the nonlinear dynamics of decay and refreshing. As memory strength gets closer to zero, the loss through decay per unit time becomes smaller, and the gain through refreshing per unit time becomes larger, until they reach equilibrium where the amount of strength lost during one operation is equal to the amount gained during one free-time interval. This analysis implies that the intuition of Barrouillet et al. (2004) is approximately correct under certain circumstances: Once enough operations have been completed so that memory strength has come close to the equilibrium for the given level of cognitive load, adding further operations with the same cognitive load has virtually no effect on memory strength. The equilibrium is reached after few operations when cognitive load is low, but requires more operations when cognitive load is high, as can be seen by comparing the two panels in the top row of Fig. 7. This is the reason why the number of operations matters most at high cognitive load, and produces the largest drop in memory performance between one and four operations, and relatively little further loss between four and eight operations.

It is important to note that the analysis of the dynamics in Fig. 7 reveals the inevitability of a shift in the point of equilibrium with cognitive load. That is, there can be no fixed set of parameter values that could create an

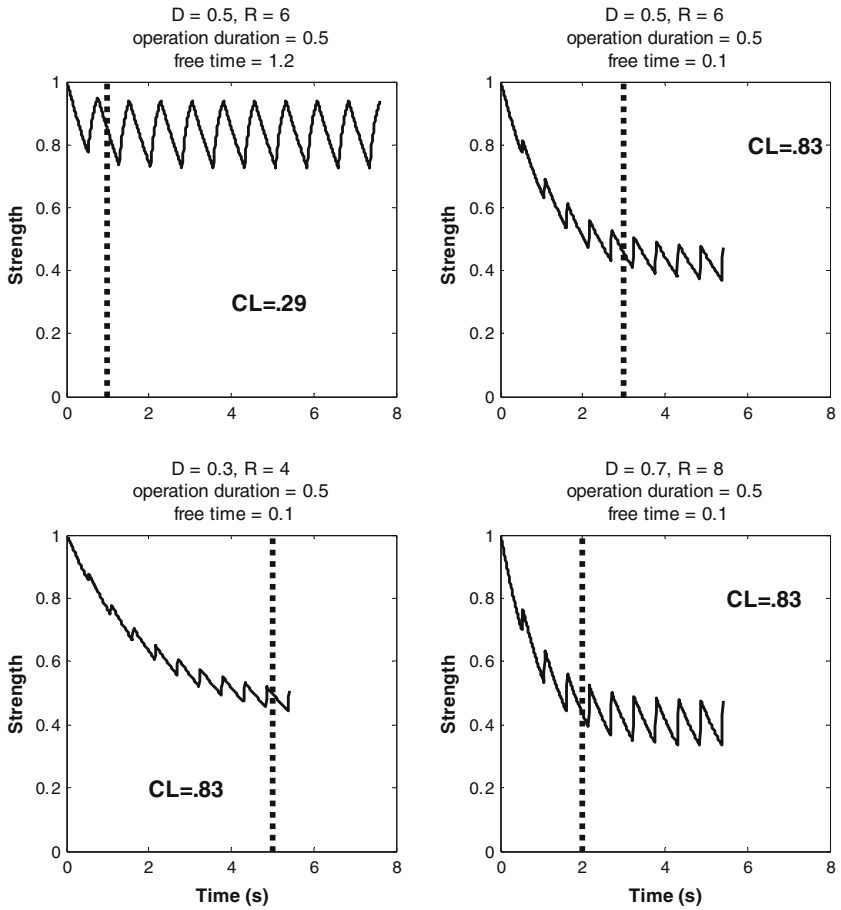

Fig. 7 Development of memory strength over time within a burst of distracting operations, each followed by a period of free time (plotted using a 10 ms time base). Strength decays during operations and is restored through refreshing during free times. Each panel shows a combination of operation duration and free time, which together determine cognitive load (indicated by "*CL*" in each panel), together with decay rate (*D*) and refreshing rate (*R*). When cognitive load is low (*top left panel*), refreshing fully compensates decay, and the number of operation has hardly any effect. When cognitive load is high (*remaining three panels*), refreshing initially fails to fully compensate decay. Eventually an equilibrium is reached, so that adding further operations does not affect strength any more. When decay rate and refreshing rate are large (*bottom right panel*), the equilibrium is reached earlier than when they are both small (*bottom left panel*). Therefore, the combination of smaller decay with smaller rate implies a larger effect of the number of operations

equilibrium—and hence a null effect of adding further operations—at the same point at all cognitive loads. This runs counter to the intuitive predictions that have been derived from the verbal model that regardless of cognitive load the number of operations has no effect (Barrouillet et al., 2004).

Discussion of Simulation 1

Simulation 1 demonstrates that our computational implementation, TBRS*, generates the main prediction of the verbal TBRS: Memory performance, measured as span or as proportion correct, declines in an approximately linear fashion with increasing cognitive load. This finding increases confidence that our implementation accurately reflects not only the meaning of the verbal theory, but also the intentions of its authors. Successful instantiation of a verbal model is far from trivial: As mentioned before, the meaning of a verbal theory—that is, the mechanisms described—leaves open many possibilities for a computational implementation (Lewandowsky, 1993). Most of these implementations, however, do not reflect the intentions of the authors, in that they generate predictions that are at odds with the intuitive predictions derived by the authors (as well as, typically, the data; see Lewandowsky & Farrell, in press, Chapter 2).

There is no guarantee that the meaning of a verbal theory is compatible with the predictions that the authors derive from it. One goal of a computational implementation of a verbal theory is to investigate whether there is a model of the theory that actually produces those predictions. In this regard Simulation 1 provides some comfort by showing that there is at least one model of the TBRS theory that generates its most important, and empirically most often confirmed, prediction.

The comfort is not complete, however, because our simulation also pointed to an instance where a prediction generated by the simulation deviates from that derived from

the verbal theory by its authors: At high levels of cognitive load, our simulation showed a detrimental effect of the number of processing operations on memory, whereas Barrouillet et al. (2004) predicted no such effect. This discrepancy could be interpreted as a failure or as a success of our modeling effort. It would be a failure if we simply missed an implementation that does predict no effect of the number of operations (as well as a decline of span with cognitive load). It is a success, however, if the modeling uncovers principled reasons for why no such model can exist. We have shown above that there is a principled reason why the combination of decay and refreshing must lead to more forgetting over longer bursts of operations when cognitive load is relatively high. Therefore, the discrepancy between the prediction of Barrouillet et al. (2004) and the prediction generated by our simulation is a discovery about what the verbal assumptions of the TBRS imply; we discuss this discovery vis-à-vis some recent data below in the context of Simulation 7.

Simulation 2: Experiment 7 of Barrouillet et al. (2004)

Our second simulation used the same model and largely the same parameter values as Simulation 1 to reproduce the data from a key experiment demonstrating the effect of cognitive load on span. Barrouillet et al. (2004, Experiment 7) measured complex span with one of two processing tasks, saying aloud the syllable "ba" or reading aloud numerals. All processing tasks were computer paced by presenting the syllable "ba" or the numeral to be read at an even pace on the screen. For reading numerals span, nine levels of cognitive load were created by crossing three levels of processing time per burst (6, 8, or 10 s) with three levels of number of operations per burst (4, 8, or 12 digits), resulting in 12 different reading paces with the available time per numeral ranging from 2.5 s to 0.5 s. For baba span, only three levels of load were created: four "ba" in 10 s, 8 "ba" in 8 s, or 12 "ba" in 6 s. Span for recall of consonant lists was the dependent variable. The results, reproduced in Fig. 8 (left-hand panel), show a roughly linear decline of span with increasing rate of processing steps per time, with a steeper slope for reading numerals than for saying "ba."

Barrouillet et al. (2004) did not compute actual cognitive load, as defined in later writing by those authors, but rather used the number of processing operations per time as the predictor of span. The two processing operations, reading a numeral and saying "ba," are likely to differ in duration, and this difference could explain the difference in the span-over-load functions for the two tasks. For computing cognitive load we need estimates for the processing durations. In their later work, Barrouillet and colleagues measured the time for reading aloud digits (Barrouillet et al., 2007), obtaining a mean duration of 424 ms. Measurement of the duration of saying "ba" in the second author's laboratory yielded a mean duration of 260 ms. It is not certain that these measures of speaking duration accurately reflect the duration for which each processing step occupies the attentional bottleneck. We nevertheless use them as estimates for the processing duration in Simulation 2 for two reasons. First, when we plotted the data of Experiment 7 in Barrouillet et al. (2004) as a function of cognitive load, computed on our estimates of processing duration (Lewandowsky, Geiger, Morrell, & Oberauer, 2010a), reading-numerals span and baba span fell on a single regression line, as they should according to the TBRS (Fig. 8, right-hand panel). Note that this replotted figure shows the data in the format favored by Barrouillet and colleagues in their more recent writing (e.g., Barrouillet et al., 2007). Our duration estimates thus facilitate a pleasingly simple explanation of the data by our computational model—unlike previously thought, seemingly all types of processing task fall onto the same cognitive load function. Second, our demonstration does not depend on the precise values for the processing durations – if the true durations are smaller than our estimates, that could easily be compensated by, for instance, increasing decay rate, reducing the encoding rate

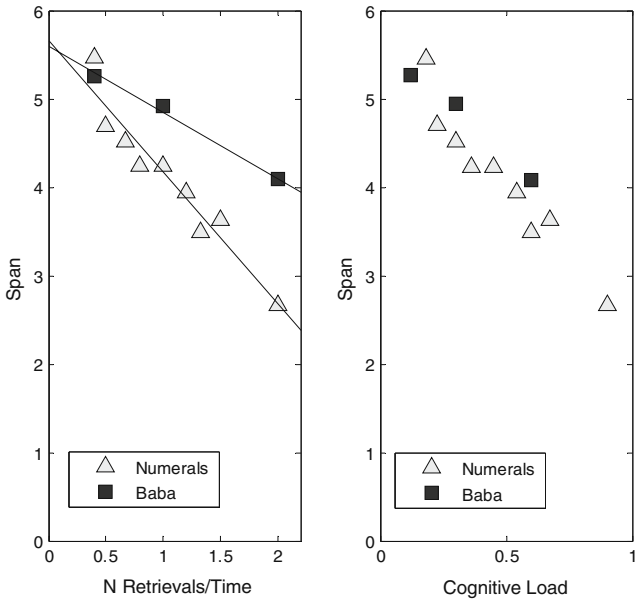

Fig. 8 Data from Experiment 7 of Barrouillet et al. (2004). The *left panel* shows span as a function of number of operations per time (from Barrouillet et al., 2004). The *right-hand panel* re-plots the same data as a function of cognitive load (from Lewandowsky et al., 2010a). The computation of cognitive load is based on empirical estimates of the duration of digit reading (424 ms, Barrouillet et al., 2007) and of uttering the syllable "ba" (260 ms, own data). The *left panel* is based on data from Barrouillet et al. (2004). Time constraints and resource sharing in adults' working memory spans. *Journal of Experimental Psychology: General, 133*, 83-100, published by the American Psychological Association, reprinted with permission

for refreshing, or assuming longer switch costs between processing operations and refreshing.

The paradigm used for Experiment 7 in Barrouillet et al. (2004) differs in one important regard from that assumed for Simulation 1: Barrouillet et al. (2004) controlled the time available for each processing operation rather than the free time after each operation was completed. Thus, the free time available for refreshing was the total time minus the operation duration. We changed the simulation accordingly, first computing the duration of each operation based on a random choice of the momentary processing rate r and on the criterion τ_{op}, and then computing the free time following that operation as the experimentally given time minus the operation duration (or zero if the latter exceeded the former).

For Simulation 2 we maintained all parameter values of Simulation 1 with the exception of the rate parameter R, which determines the speed at which learning strength at encoding and refreshing increases over time. R was lowered from 6 to 5 for this simulation, because with $R = 6$ span increases too steeply with decreasing load. Figure 9 shows the results. The model reproduced the data well, with one noticeable exception: The model predicts a deceleration of the span-over-load function as cognitive load decreases; this was not observed in the experiment. This prediction arises because low cognitive load is characterized by very long free time (i.e., more than 2 s per operation at the lowest level of load, where 2.5 s are available for reading each numeral, which takes < 0.5 s), and as shown above, the model produces diminishing returns from increasing free time. At present, it is not clear whether this discrepancy reflects a limitation of the model or of the data. It is possible that the model makes the wrong prediction at very low levels of cognitive load or that the experiments carried out so far were not sensitive enough to pick up the flattening of the span-over-load function at very low cognitive load.

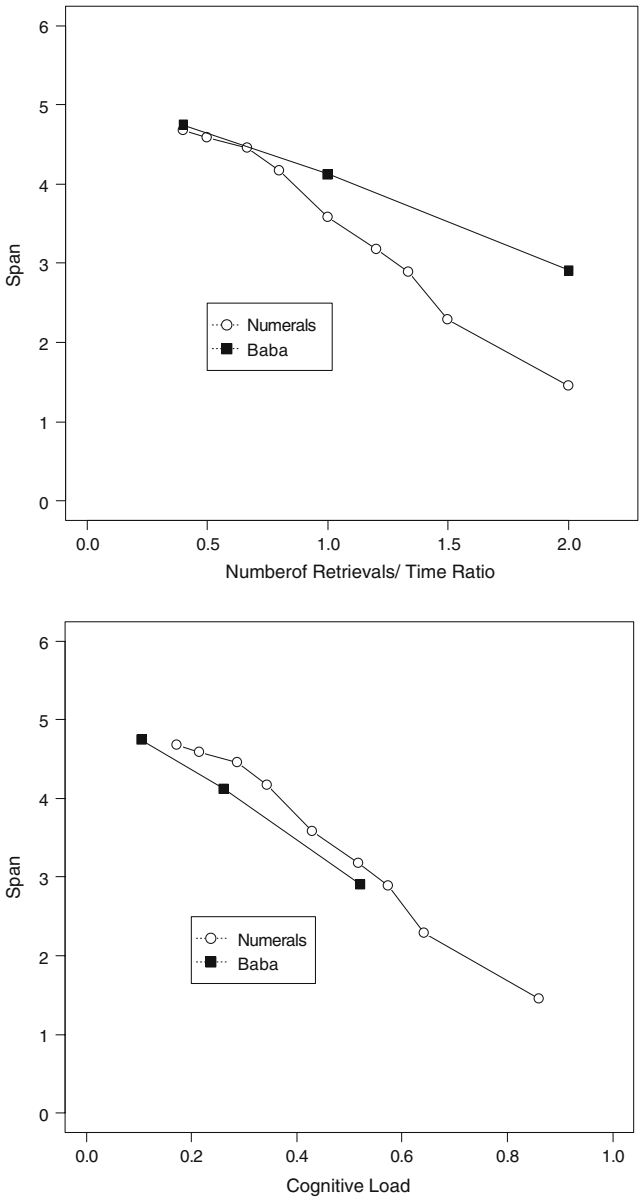

Fig. 9 Results of Simulation 2, reproducing Experiment 7 of Barrouillet et al. (2004) in TBRS*. The *top panel* plots the data as a function of number of operations per time (as in Barrouillet et al., 2004); the *bottom panel* plots the same data as a function of cognitive load (as in Lewandowsky et al., 2010a).

Serial position curves and types of errors

We next turn to a more fine-grained analysis of the model's behavior. Researchers of serial recall have routinely analyzed their data for serial position curves and types of errors. The principal results, such as extended primacy accompanied by small (i.e., 1- or 2-item) recency, are highly reproducible across experiments and have turned into widely accepted benchmarks for computational models of serial recall. Therefore, a model of the TBRS should aspire to explaining serial-position data and error-type data as well.

Serial position curves

Serial position curves plot accuracy as a function of the serial position of the to-be-recalled item. A typical finding from immediate forward serial (i.e., simple span) is an extended primacy effect —i.e., enhanced performance for items that appeared early on the list—together with a recency effect typically confined to the last one or two items. Hardly any analyses of serial-position effects have been published for complex span, but as we will show below, the shape of the curve is very similar to that for simple span.

Figure 10 shows serial-position curves for seven-item lists from Simulation 1 for simple span (0 operations) and for complex span with different numbers of operations,

averaged across levels of cognitive load. These curves reproduce the typical finding of extended primacy and a less extended recency effect. TBRS* generates the recency effect through decay: Because recall is faster than the pace of item presentation, the last list items have the least time to decay. The model generates a primacy gradient through cumulative refreshing: Refreshing starts at the beginning and continues until it is interrupted by another demand on the attentional bottleneck; this interruption usually occurs midway through the list, so that list-initial items receive more refreshing. The role of refreshing in generating primacy can be seen directly when free time is reduced to zero, as shown in Fig. 11.

The top panel of Fig. 11 displays serial position curves for the complex span data from Simulation 1, broken down by available free time, averaging across operation duration and number of operations (1, 4, or 8). When free time is zero, the primacy effect virtually disappears. With just 100 ms of free time between operations, a strong primacy effect emerges, but it hardly extends beyond the first item. As free time increases, the primacy effect extends further into the list. Note that when free time is zero, it does not mean that no refreshing occurs at all – the model still refreshes in the residual item presentation time after encoding of an item has reached asymptote, but it has no free time in between operations within a burst. As a consequence, memory decays unbridled throughout the burst, which drastically diminishes the strength of the items encoded so far. This, in itself, would not prevent them from being recalled, but once the next list item is encoded, it is relatively much stronger than the preceding items, so that when one of the earlier items is cued for retrieval, the most recently encoded item is likely to be retrieved instead. In other words, early list items must be refreshed fairly continuously, lest their strength drops too far below that of more recently encoded items, thus rendering the earlier items irrecoverable.

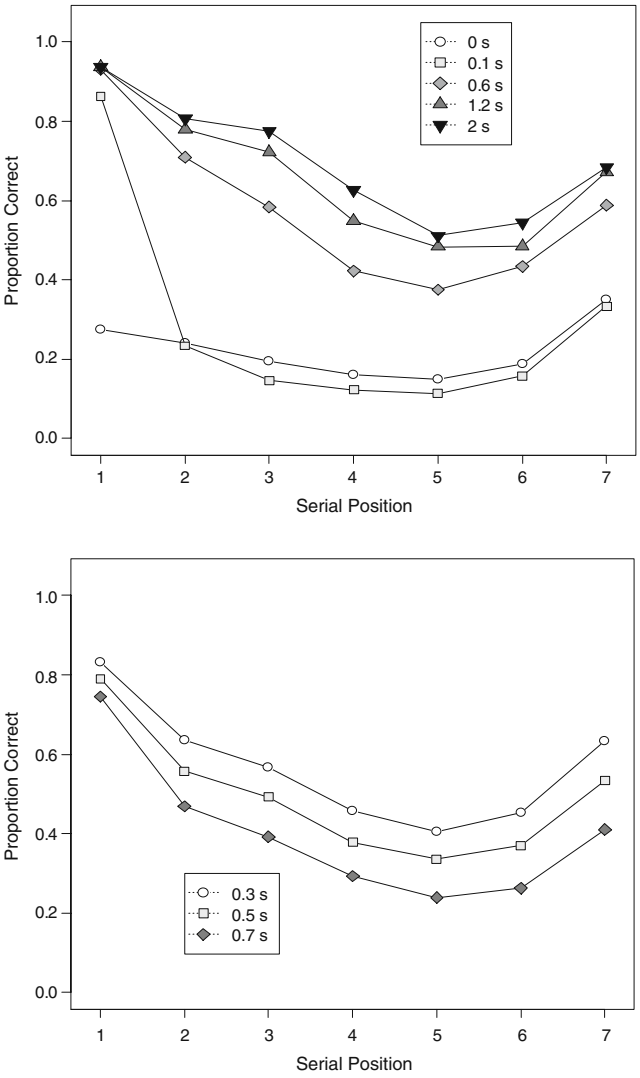

Fig. 11 Serial-position curves for seven-item lists as a function of free time (*top*) and of operation duration (*bottom*); data from Simulation 1

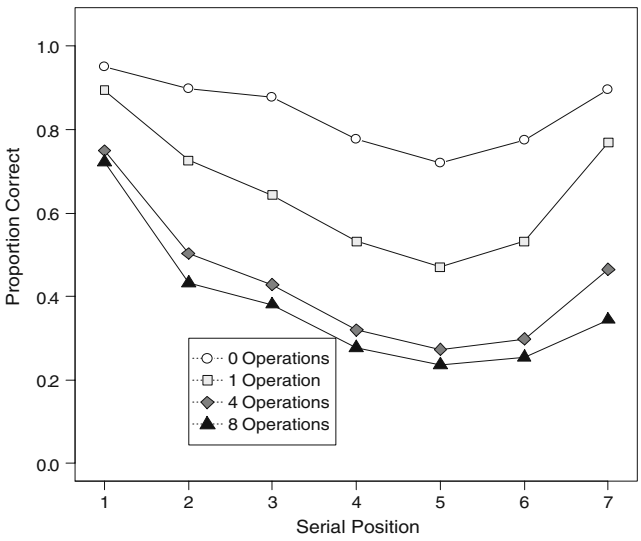

Fig. 10 Serial-position curves for seven-item lists as a function of number of distractor operations; data from Simulation 1

The bottom panel of Fig. 11 shows serial position curves by operation duration. A comparison of Fig. 10 and the two panels of Fig. 11 reveals that TBRS* makes different predictions for how manipulations of the number of operations, of the available free time, and of the duration of operations affect the serial position curve. Increasing the number of operations impairs memory primarily in the middle of the list; decreasing free time reduces and eventually eliminates the primacy effect, and increasing

operation duration affects all list items to about the same degree.

Whereas serial position effects are routinely analyzed in studies of serial recall, serial-position data from complex span tasks are scarce, with the data typically being reported in aggregate fashion (e.g., as span scores). The most thorough analysis so far has been presented by Unsworth and Engle (2006). Their data come from the large-scale study by Kane and colleagues (Kane et al., 2004). In that study, participants were asked to recall items in correct order by writing each item into a slot on an answer sheet. The experimenters did not control the temporal order in which participants wrote down their responses (Kane, personal communication, October 15, 2008). Therefore, it is possible that people deviated from recall in forward order, for instance by first writing the last item into the last slot, a common pattern in serial retrieval (Lewandowsky, Brown, & Thomas, 2009). The lack of control of output order is unfortunate because output order has a large influence on accuracy (Cowan, Saults, Elliott, & Moreno, 2002; Oberauer, 2003), and therefore any deviation from forward recall distorts the serial position curve. This consideration speaks against using the data from Unsworth and Engle (2006) as benchmark data for evaluating models of complex span. Instead, we present in Fig. 12 data from our own laboratory that were obtained with a computerized test battery for working memory (Lewandowsky, Oberauer, Yang, & Ecker, 2010b), which includes reading span and operation span. Our span tasks are well suited for comparison with the predictions of TBRS* because they use fixed, computer-controlled times for each processing burst (though not for individual operations within a burst), and they force participants to enter items in forward order. The top panel of Fig. 12 shows data for reading span and the bottom panel for operation span; both are taken from Experiment 2 in Lewandowsky et al. (2010b). Comparison of the data (for list length 7) with the predicted form of the serial position curve (Fig. 10) shows qualitative agreement; the model, like the data, shows extensive primacy and limited recency.

Unfortunately, the data in Fig. 12 do not include a manipulation of the number of operations, operation duration, or free time. We will discuss an experiment manipulating the number of operations below. We are not aware of serial-position data from an experiment manipulating operation duration and free time independently, so the corresponding predictions of TBRS* in Figs. 10 and 11 remain to be tested.

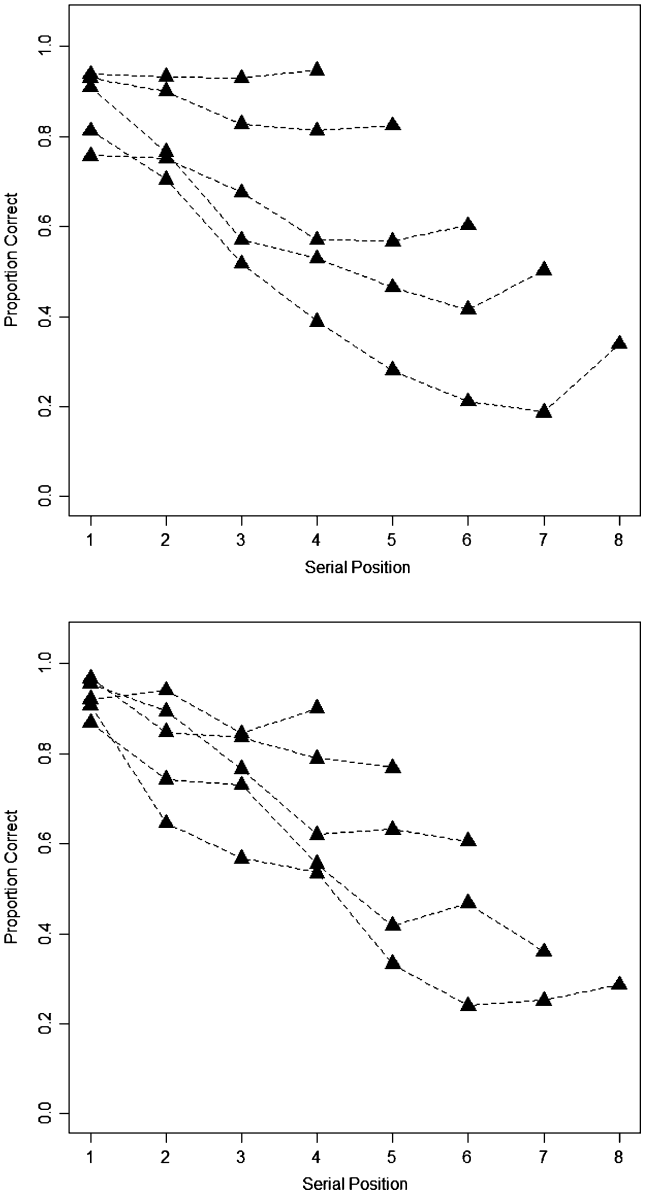

Fig. 12 Serial position curves for an operation span task (*top panel*) and for a sentence span task (*bottom panel*) for various list lengths. Data are taken from Experiment 2 of Lewandowsky et al. (2010b) and involve keyboard recall of lists of consonants, each of which was preceded by a single arithmetic operation (e.g., 3 + 2 = 5) or a single sentence (e.g., "All working memory experiments are clever") that had to be verified (true or false indicated by keypress)

Error types

Two broad categories of errors are distinguished in serial recall, item errors and order errors. A response is regarded as an order error if at a given output position a list item from a different position is recalled. These errors are also referred to as transpositions. A response is regarded as an item error if at a given output position no item from any list position is recalled. Item errors can be either omissions (if participants are allowed to skip an item) or extralist intrusions, that is, items that were not presented on the current list.

Figure 13 displays the predicted proportion of order errors, omissions, and extralist intrusions as a function of

serial position. These data are from all complex span conditions of Simulation 1, averaging across number of operations and levels of cognitive load. The corresponding behavioral data are shown in Fig. 14, which are taken from Experiment 2 in Lewandowsky et al. (2010b)–with reading and operations span in the top and bottom panel, respectively–for a seven-item list from the same complex span experiment that produced the serial position curves in Fig. 12. The experiment did not permit omissions so all item errors were intrusion errors. TBRS* accurately predicts the increase of intrusion errors over serial position.

Turning to order errors, Fig. 15 shows the empirical transposition gradients from the same study (Experiment 2 in Lewandowsky et al., 2010b), and Fig. 16 shows the corresponding predictions from TBRS* (Simulation 1). In the figures, each item is identified by its input position, and the corresponding line in the graph shows the probability of that item being recalled at each output position. These plots illustrate the transposition profiles for a seven-item list in a complex span task averaging across number of operations (1, 4, or 8). A comparison of Figs. 15 and 16 clarifies that TBRS* captures the transposition gradients reasonably well. In particular, when an item is recalled in the wrong output position (i.e., a transposition error), it tends to be recalled in a nearby position. This is a common pattern in serial recall; our model generates it in the same way as many previous models (e.g., G. D. A. Brown et al., 2000; Burgess & Hitch, 1999; Henson, 1998b), through position representations that overlap more the closer they are to each other. However, it is also clear from Fig. 16 that the model predicts more transpositions than were observed in the data, especially at central serial positions.

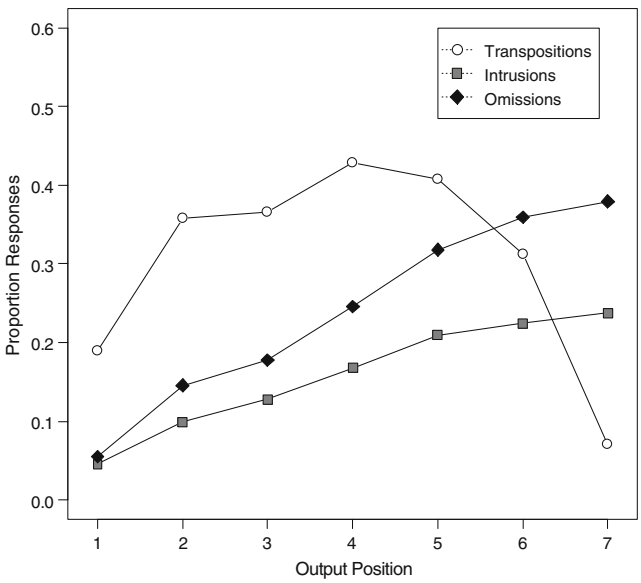

Fig. 13 Types of errors generated by TBRS* as a function of serial position. Data are from Simulation 1

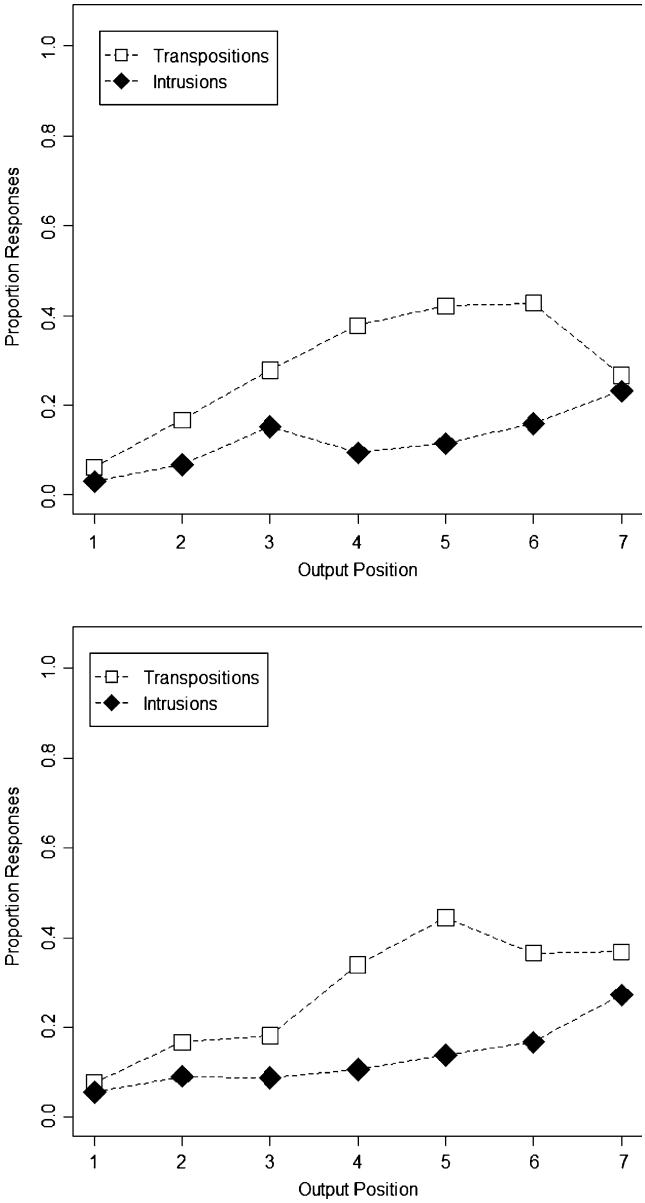

Fig 14 Types of errors as a function of serial position; data from Experiment 2 in Lewandowsky et al. (2010b) for sentence span (*top*) and operation span (*bottom*)

TBRS^0 : decay and the relativity of memory strength

Before we move on to show the wider applicability of TBRS*, we examine the mechanisms of forgetting in our instantiations. The notion of decay is closely associated to the image of a memory trace that fades away over time until it becomes irrecoverable. This image is misleading because it focuses on the fate of an individual representation,

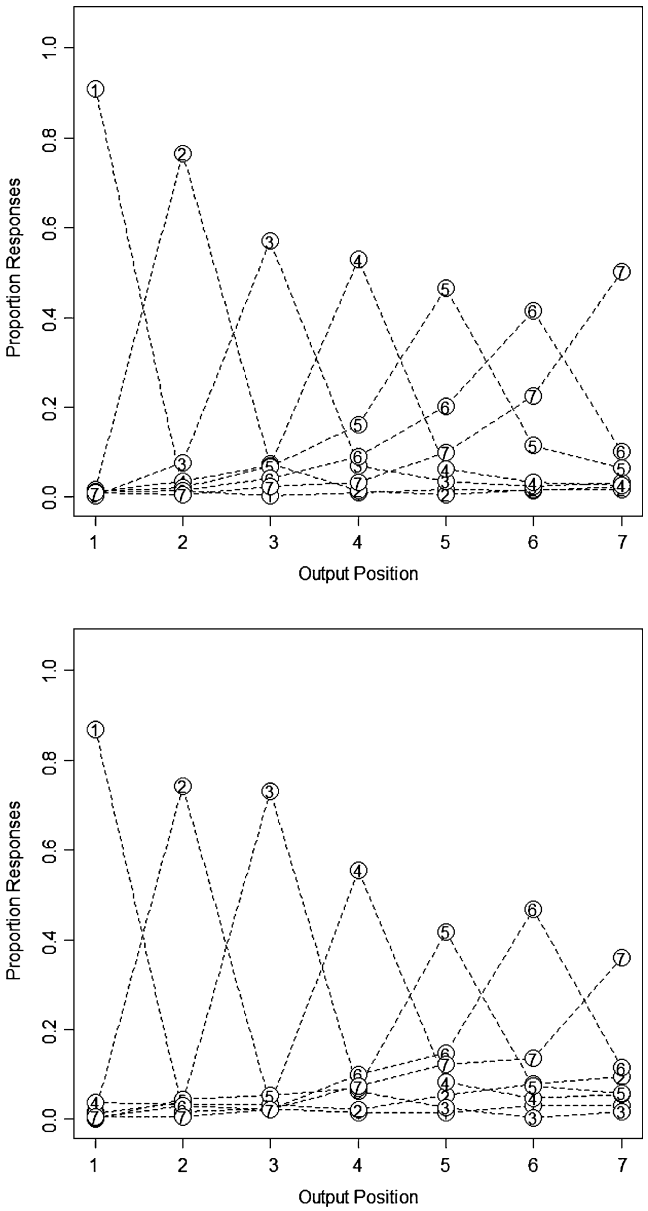

Fig. 15 Transposition profiles for seven-item lists from Experiment 2 in Lewandowsky et al. (2010b). Data are from sentence span (*top*) and operation span (*bottom*). Input positions are represented by numbers within plotting symbols

ignoring the role of other representations that interact with the target representation at retrieval. Decades of research on human memory have demonstrated time and again that retrieval depends not so much on the characteristics of the memory representation to be retrieved but on its relation to other representations that act as cues or competitors (Surprenant & Neath, 2009). This principle of relativity is true also for our instantiation of TBRS. In TBRS*, the success of an attempt to retrieve a particular item depends on both that item's absolute strength and on its relative strength compared to other list and extralist items. Absolute strength is important because an item is recalled only if its activation, after cueing by a position, exceeds the retrieval threshold, and relative strength is important because an item is recalled only if its activation exceeds that of all other recall candidates.

To investigate the roles of absolute and relative strength, we created a stripped-down version of the model, which we call TBRS^0, in which we set the retrieval threshold θ and the retrieval noise σ to zero. Setting the threshold to zero implies that no item can ever decay below the threshold. Setting noise to zero implies that no extra-list item can ever acquire non-zero activation, because they are not associated to any unit in the position layer and thus receive no input from the positional cue. As a consequence, TBRS^0 cannot commit any item errors. Thus, during any recall attempt in TBRS^0, the item with the highest activation in the item layer (after cuing) is recalled, even if that activation is minuscule, and the item is guaranteed to be from the currently studied list. Simulation 3 explored the implications of this stripped-down version, TBRS^0, and the results are presented in Figs. 17 and 18.

Figure 17 shows that TBRS^0 can produce plausible cognitive-load functions (top panel) as well as serial-position curves (bottom panel). Thus, the most important predictions of TBRS are generated by a model version in which no memory trace, looked at individually, ever decays below a threshold that renders it absolutely irrecoverable. Figure 18 provides a closer look at order errors produced by TBRS^0. The figure again shows the probability of each item

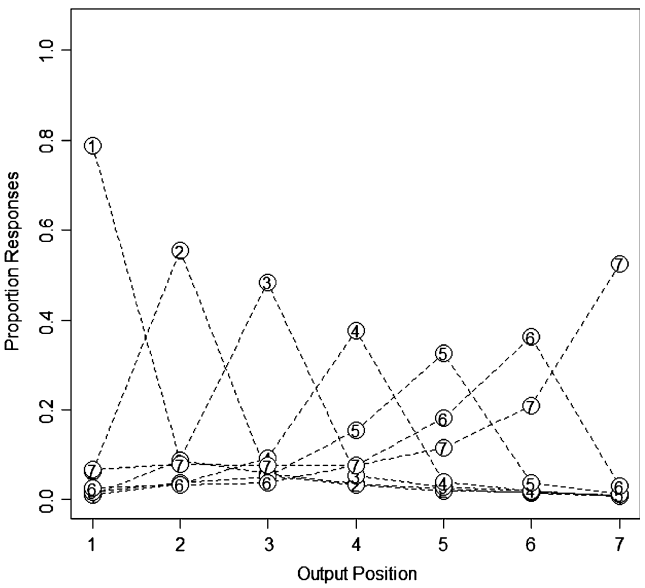

Fig. 16 Transposition profiles predicted by TBRS*. Data are from Simulation 1, averaged across numbers of operations (1, 4, or 8), and across all levels of cognitive load. Input positions are represented by numbers within plotting symbols

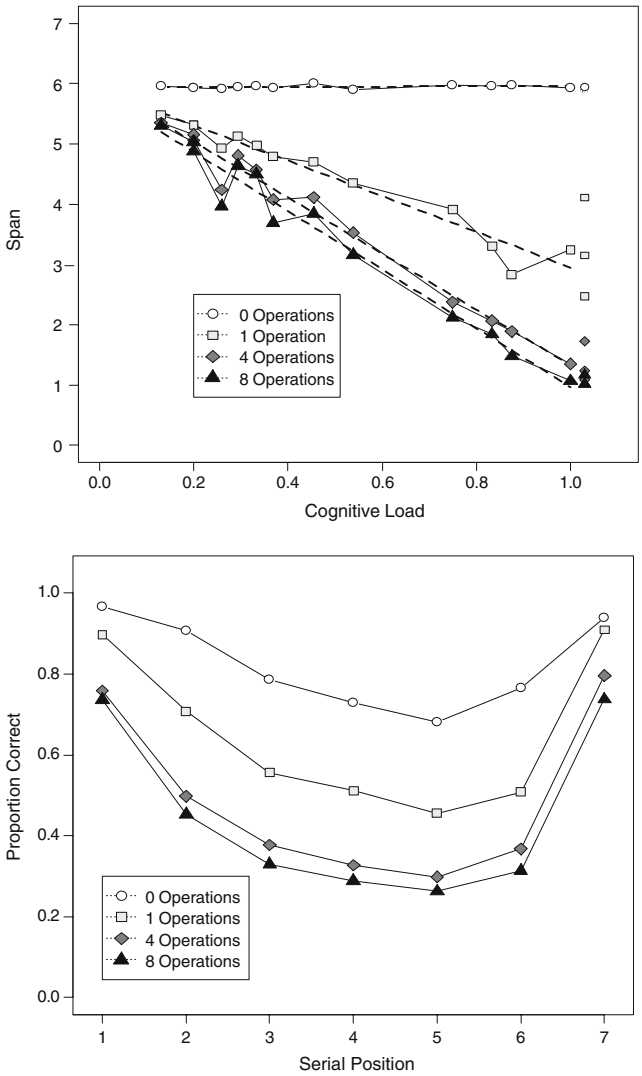

Fig. 17 Predicted cognitive-load functions from TBRS^0 (*top panel*) and predicted serial-position curves from TBRS^0 for seven-item lists as a function of number of distractor operations (*bottom panel*)

being recalled at each output position. These plots illustrate the transposition profiles for a seven-item list in a complex span task averaging across number of operations (1, 4, or 8), for a relatively long free time (1.2 s) in the top panel and for zero free time in the bottom panel. The plots show one noteworthy feature of the stripped-down model: There is a strong tendency for later list items to be recalled too early, and this tendency is accentuated as free time is reduced. These anticipation errors arise in the model because decay creates a recency gradient of strength. Because positional cues overlap with other nearby position representations, cuing for recall by a position representation will activate not only the item in that position but also, to a lesser degree, items in neighboring positions. The activation that item n receives at retrieval in the item layer, given that position m is used as cue, is a function of three variables, the degree of overlap between neighboring positional representations P, the distance between positions m and n, and the strength of association of item n to position n at the moment of retrieval, w_{nn}. Because overlap decreases exponentially with distance, the degree of activation of item n when cued with position m can be expressed as:

$$a_{n|m} = w_{nn} \cdot P^{|m-n|}. \tag{9}$$

Thus, when item m is cued for retrieval, item n in a later position can achieve higher activation at retrieval than item

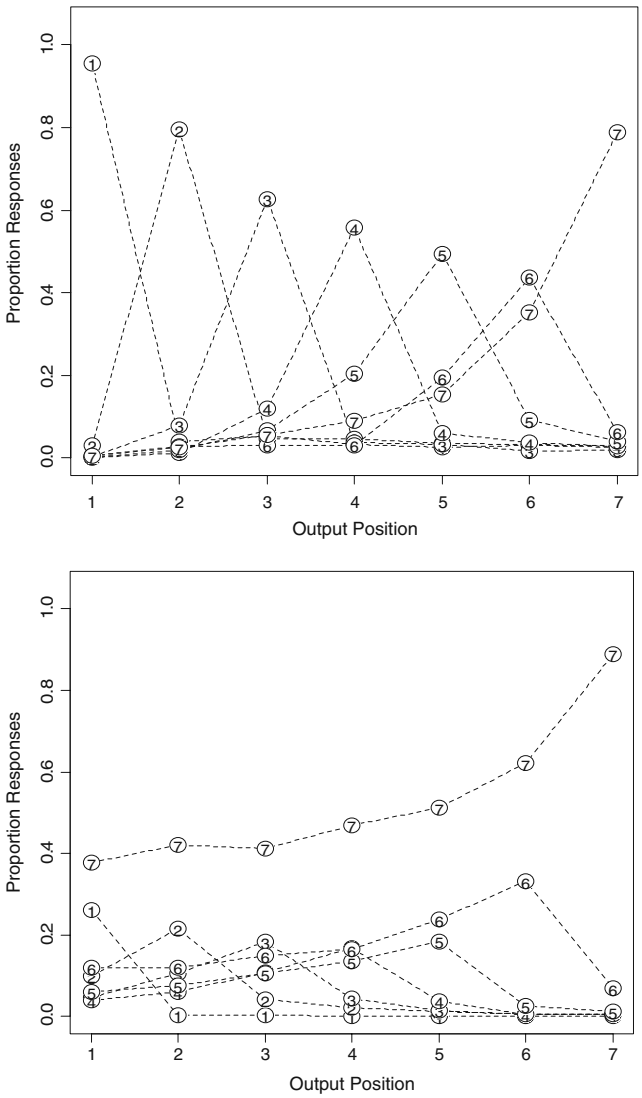

Fig. 18 Transposition profiles for a seven-item list of a complex span task produced by TBRS^0. Data are from Simulation 3 with 1.2 s of free time (*top panel*) and zero free time (*bottom panel*). Each *line* represents an item identified by its input position (number in the data points), and the data points reflect the probability of recalling that item in each output position

m if the strength of item n (w_{nn}) is much larger than that of item m (w_{mm}), so that the difference in strength exceeds the discounting by positional distance, $P^{|m-n|}$. This is how anticipation errors arise from a strong recency gradient as a consequence of decay. Refreshing serves to battle anticipation errors by generating a primacy gradient of strength, and therefore the list-initial items are largely protected from anticipation errors when there is sufficient free time for refreshing. When no free time is available, no primacy gradient is built to stem the flood of anticipation errors, as shown in the bottom panel of Fig. 18.

This analysis of TBRS^0 raises questions about the role of decay in generating the predictions of TBRS. In TBRS^0, an error of retrieval occurs if and only if the correct item's level of activation in the item layer, when cued by its position, is surpassed by another item's activation. Thus, it is the *relative* level of activation of different items after cueing with a position that determines retrieval accuracy, not their absolute level. This means that as long as the relative levels of activation each item receives when cued for retrieval remains the same, their absolute level can assume any value without changing the model's behavior. Because the activation levels of items at retrieval is determined by the connection weights between position layer and item layer, this means that as long as the relative connection weights do not change, their absolute values do not matter. How, then, can decay, which reduces the absolute strength of connection weights, be responsible for forgetting in TBRS^0?

The answer is that decay plays an important role in causing forgetting, even in the absence of a threshold, but it can play this role only in collusion with other processes. One such process is the temporally staggered encoding of items. When items are encoded at different times, decay creates differences in strength between items: Earlier items decay for longer time than later items. Without refreshing, earlier items end up weaker than later items, creating a recency gradient of strength. When a list needs to be recalled in forward order, a recency gradient is unfortunate because later list items tend to intrude when earlier list items are cued for recall. To counteract the trend towards a recency gradient of strength, the cognitive system can engage in refreshing. Cumulative refreshing preferentially boosts the list-initial items, thus preventing them from falling behind. Refreshing evens out the relative strength of items to some degree, but not perfectly, because there is no omniscient homunculus that can direct refreshing precisely to the weakest items at any moment and boost them just enough for them to catch up. Rather, refreshing proceeds according to a constant plan – cumulative refreshing in forward order in our instantiation – regardless of the "needs" of individual items. As a consequence, refreshing ameliorates the gross unevenness between earlier and later items that arises from decay, but at the same time creates uneven memory strength itself: Some items get refreshed more often than others. Because refreshing starts with the first item, the list-initial items tend to get more refreshing than later ones, thus creating a primacy gradient on strength. The recency gradient arising from decay and the primacy gradient arising from cumulative refreshing together give rise to the U-shaped serial position curve that is typical for serial recall. Items in the middle of the list are recalled worse than those at the beginning and the end – not because they are too weak to be retrieved, but because they often lose the competition for retrieval to other items that are stronger. In the majority of cases, the stronger competitors are later list items, which are thus recalled too early, whereas earlier list items are typically suppressed after having been recalled themselves. Only after a failure to recall an earlier list item can that item become a strong competitor for recall of a later list item.

To further understand the behavior of the model's core properties, we systematically explored the parameter space. The most interesting results of this effort are summarized in Appendix B. Within the region of parameter space explored, we did not find a combination of parameter values that approximates the benchmark findings as well as, or better than, the values we used for Simulations 1 and 3. We are confident that, at least within the range of parameter values that we explored, there is no set of substantially different parameter values with which TBRS^0 (and TBRS*) can reproduce the benchmark findings equally accurately.

Having analyzed the core properties of the model's architecture, we next turn to applications of our full model, the TBRS*.

Applications

The following three simulations apply TBRS* to two experiments with the complex span paradigm and to a related paradigm, the Brown-Peterson task.

Simulation 4: An application to the development of working memory

Simulation 4 applies TBRS* to a recent experiment of Barrouillet et al. (2009). Their first experiment tested children at four age groups (8, 10, 12, and 14 years) on a complex span task in which participants had to remember letters and read aloud digits. Cognitive load was varied by presenting the digits at a pace of 0.4, 0.8, 1.2, or 2 digits per second. Span was found to decline approximately linearly with increasing cognitive load in all age groups, and the slope of the cognitive load function was steeper in older children. Barrouillet et al. took the age difference in slope as evidence that younger children did not refresh memory

as efficiently as older children, therefore taking less advantage of the free time periods at low loads. In the second experiment, Barrouillet et al. (2009) equated cognitive load between two age groups (8 and 14 years) by first measuring the average time for reading aloud a digit (622 and 489 ms in younger and older children, respectively) and then testing complex span with paces for the digit-reading component adapted to each age group's speed of digit reading. That is, three levels of cognitive load were created by presenting digits for once, twice, or four times the average digit-reading time for the child's age group. Barrouillet et al. (2009) assumed that the digit reading times reflect the times during which the attentional mechanism is engaged by digit reading. It follows that both age groups worked at cognitive-load levels of 1, 0.5, and 0.25.

With cognitive load equated, the older children still had higher spans than the younger children, but the slopes of their cognitive load functions were no longer significantly different. The authors concluded that age differences in the rate of refreshing, as reflected in the speed of digit reading, account for part but not all of the age differences in complex span performance. The remaining age differences after equating cognitive load must be attributed to another factor. Barrouillet et al. (2009) eventually identified four variables that they held responsible for developmental changes in working memory: rate of processing (i.e., reading digits), rate of refreshing, speed of decay, and attentional capacity. The latter is assumed to determine the strength of encoding of items, as well as the strength of refreshing.

Simulation 4 instantiated Experiment 2 of Barrouillet et al. (2009). We represented age differences as follows: Differences in processing rate were reflected in the measure of digit reading speed that determined the pace of digit reading for each age group, so we assume that the mean operation duration for the older children was 0.489 s, and for younger children, 0.622 s; these values were translated into the appropriate processing rates by Eq. 8a. Refreshing rate is reflected in parameter R and decay rate in parameter D. The strength of encoding is controlled by the encoding criterion τ_E. Therefore, we concentrated our efforts to reproduce the data on variations of these three parameters: We obtained good results, shown in the top panel of Fig. 19, with values of $D = 0.5$, and $\tau_E = 0.5$ for the younger children, and with $D = 0.4$, and $\tau_E = 0.55$ for the older children. Surprisingly, reducing R as a function of age turned out to be unnecessary for obtaining a good fit, and therefore we left R at its default value of 6.

One interesting outcome of this simulation is the fact that the refreshing rate was constant between age groups, which is at odds with the conclusion of the verbal theorizing offered by Barrouillet et al. (2009). To delineate that conflict, it must be borne in mind that the parameter R affects the refreshing rates but not the rates of carrying out

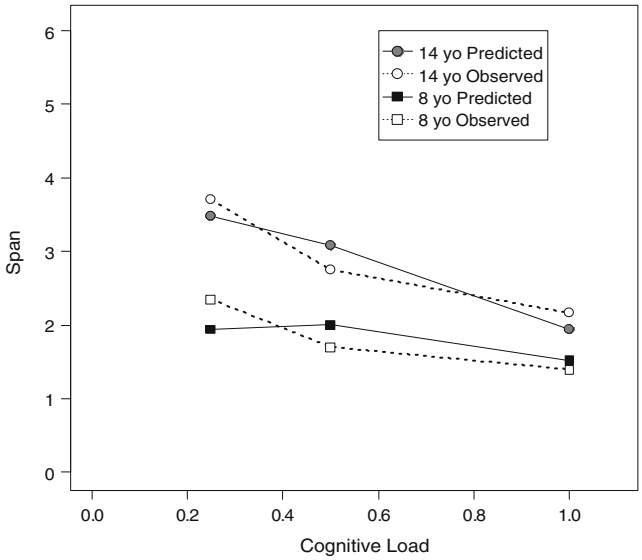

Fig. 19 Data from Experiment 2 of Barrouillet et al. (2009) with model predictions from Simulation 4

the digit-reading task—because the processing rate for operations, R_{op}, is determined solely by the independently measured processing times (see the earlier discussion surrounding Eq. 8a). Thus, even though our simulation kept R constant, it allowed for age-dependent processing rates, similar to Barrouillet et al. (2009). However, unlike Barrouillet et al., our simulation showed that refreshing rates need not change with age—only the processing rate, the decay rate and the strength of encoding must be assumed to be age dependent.

We conclude from this analysis that Barrouillet et al. (2009) identified the correct variables that need to be manipulated to explain their data – with one exception: contrary to their assumption, the data of their Experiment 2 do not compel the conclusion that refreshing rate differs between age groups.[3]

It does not follow from our modeling that refreshing rate remains invariant across age. Other findings need to be taken into account before one might reach that conclusion. Instead, what our simulations show is that the data of Experiment 2 of Barrouillet et al. (2009) are by themselves insufficient to conclude that refreshing speed changes across age, because they can be fit by the model just as well without assuming such change. It follows that the issue as to whether refreshing rate changes with age or is age-invariant remains open.

[3] Our conclusions about the development of working memory come with a caveat: Barrouillet et al. (2009) did not report how often children omitted reading a digit in their Experiment 2. If that omission rate is non-negligible, the actual cognitive load would be lower than their estimate that we used for the simulations.

Simulations 5 and 6: Cognitive load in the Brown-Peterson paradigm

So far we have looked at how the model behaves in the complex span paradigm for which the TBRS was initially developed. Here we investigate how TBRS* applies to a related paradigm, the Brown-Peterson paradigm (J. Brown, 1958; Peterson & Peterson, 1959). The Brown-Peterson paradigm is similar to complex span in that it combines memory of short lists—typically no more than three letters or words—with a period of distractor activity—typically counting backwards aloud by threes from a random three-digit number. The main difference is that in the Brown-Peterson task the distractor activity consists of a single burst that follows presentation of the whole list. For three-item lists, the typical Brown-Peterson result consists of initially rapid forgetting as the distractor period is increased (to approximately 10 s), followed by a leveling off and a reduced forgetting rate until asymptote is reached at around 15-18 s.

It is plausible to assume that decay and refreshing operate in the Brown-Peterson paradigm in the same way as in complex span; indeed, recent research by Barrouillet and colleagues has used variants of the Brown-Peterson paradigm (Liefooghe, Barrouillet, Vandierendonck, & Camos, 2008; Vergauwe et al., 2009). Therefore, Simulation 5 first examines those existing data before Simulation 6 generates new predictions from TBRS* for a broader examination of the Brown-Peterson paradigm.

Simulation 5 sought to model the results of Experiment 4 of Liefooghe et al. (2008). In their study, subjects studied lists of 4, 6, or 8 consonants (for 1.8 s each). Presentation of the list was followed by eight processing stimuli (for 1.2 s each). Each processing stimulus involved a single digit that had to be classified either by magnitude (less than or greater than 5) or parity (odd or even), as cued by the color in which the digit was presented. Cognitive load was manipulated in two ways: by varying the number of task switches (from parity to magnitude or vice versa) within the sequence of 8 stimuli or by physically degrading the stimuli to impair identification. Liefooghe et al. argued that both manipulations increase the duration of attentional capture. The results of the study are shown in Table 5 together with the predictions obtained from TBRS*. The simulation instantiated the methodology of Liefooghe et al. by presenting all memoranda in a row before administering the processing task, with the simulated operation duration being the average of the total processing times reported by Liefooghe et al. for each experimental condition.

The table shows that TBRS* captured the main features in the data with considerable quantitative precision. Note that the decay rate (*D*) had to be lowered to .35 (from its usual value of .5; see Table 3) for the model to be able to capture the data. With that single change in parameter values, TBRS* was able to accommodate the only study known to us that manipulated cognitive load during the retention interval (the study by Vergauwe et al., 2009, used a related procedure but only presented a single item and used recognition rather than recall.)

Table 5 Data and predictions from TBRS* (Simulation 5) for Experiment 4 of Liefooghe et al. (2008)

	List length			*M*
	4	6	8	
Data				
Low-switch	.93	.75	.62	*.77*
High-switch	.90	.71	.56	*.72*
Low-switch degraded	.91	.73	.54	*.73*
Predictions				
Low-switch	.92	.86	.63	*.80*
High-switch	.91	.76	.57	*.75*
Low-switch degraded	.90	.76	.56	*.74*

We next explored the behavior of TBRS in the Brown-Peterson paradigm more broadly. Simulation 6 tested serial recall of five items presented for 1.5 s each. Each list was followed by 0, 1, 4, 8, or 16 distractor operations. The number of operations was crossed with two levels of operation duration (0.3 or 0.6 s) and three levels of free time (0, 0.5, or 1 s) to generate six levels of cognitive load. Figure 20 shows the results in the format commonly used to present results from the Brown-Peterson paradigm. TBRS* reproduced the standard forgetting curve: proportion correct recall decreased as a function of number of distractors. The bottom panel contains a thumbnail sketch of representative data, taken from Murdock (1961). The slope of the simulated forgetting curve is strongly modulated by operation duration and free time, confirming a marked effect of cognitive load. At the lowest level of load, the model predicts no forgetting at all.

In summary, without introducing any further modifications, our instantiation of the TBRS can handle the existing data concerning the role of cognitive load in the Brown-Peterson paradigm, and it generates quite plausible predictions for future research in this paradigm. Having shown the basic feasibility and wide applicability of our instantiation of TBRS, we conclude by discussing some data that clearly challenge the basic assumptions of the theory.

Simulation 7: A challenge for the TBRS: constant vs. changing distractors

Our final simulation addresses a challenge for the TBRS, and for decay-based models of working memory in general. This challenge arises from a series of experiments we carried out to adjudicate between decay and interference as

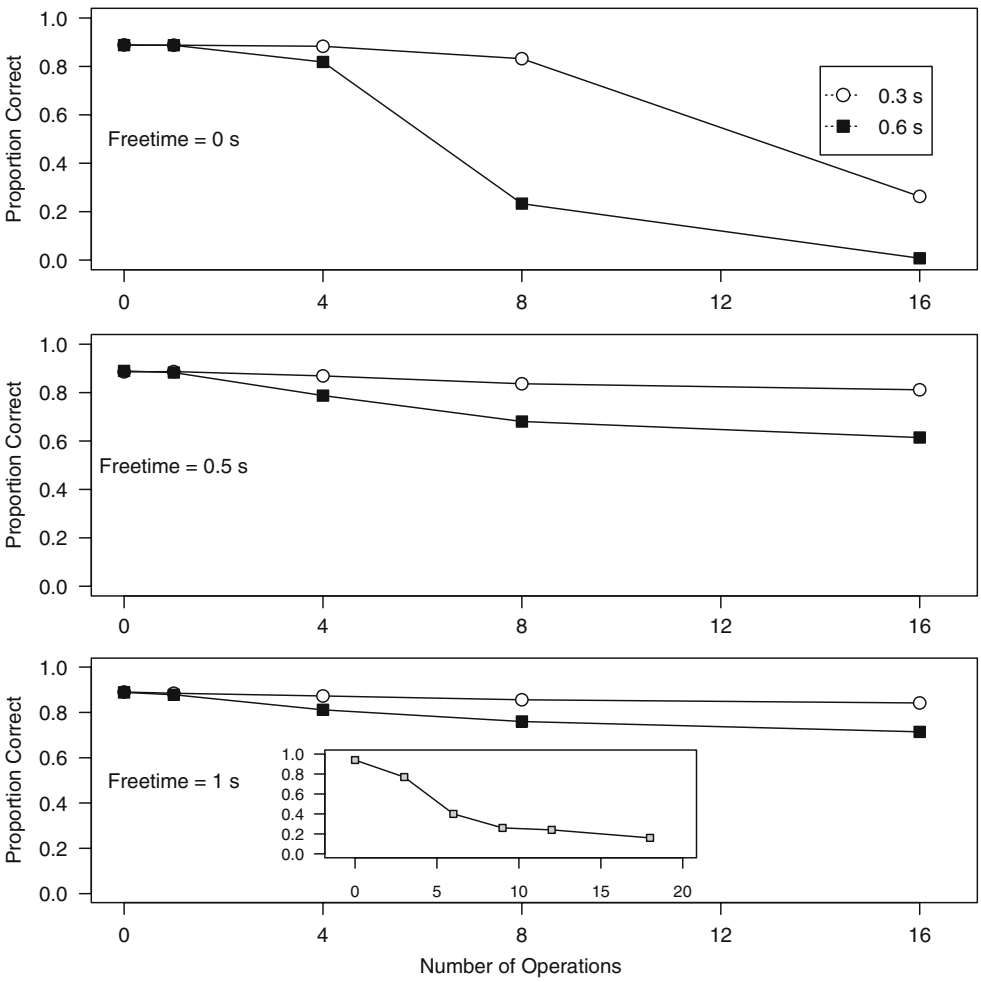

Fig. 20 Model predictions for the Brown-Peterson task at six levels of cognitive load (Simulation 6). The time for memory restoration after each operation is varied between panels, and two different operation durations are represented by the two parameters in each panel. The *thumbnail insert* in the *bottom panel* shows typical behavioral data for consonant trigrams as memoranda (taken from Murdock, 1961)

causes of forgetting. We investigate whether and under which conditions the TBRS can account for the results of these experiments.

We used a complex span procedure in which participants remembered lists of letters, and the processing task consisted of speaking distractor words aloud. In the original experiment, distractors were inserted between recall attempts (Lewandowsky, Duncan, & Brown, 2004); in later experiments we also inserted distractor words to follow each item during encoding, as in the standard complex span paradigm (Oberauer & Lewandowsky, 2008). When the distractor word was the same throughout a trial, it made no difference for recall whether that word was spoken once or three times after each item during encoding. Likewise, the number of distractor words preceding each item at retrieval had only a negligible effect on memory (Lewandowsky et al., 2004; Oberauer & Lewandowsky, 2008). In contrast, when the words were all different, memory was worse with three distractors per memory item than with a single distractor (Lewandowsky et al., 2010a; Lewandowsky, Geiger, & Oberauer, 2008). In other words, whether the number of operations affects memory was modulated by the similarity between successive distractors that accompanied each item. This pattern of effects is just as predicted from a model based on interference, rather than decay, as the cause of forgetting, namely the Serial-Order in a Box (SOB) model (Lewandowsky & Farrell, 2008). The findings are not easily accommodated by the TBRS for two reasons. First, for the reasons noted earlier (see the discussion surrounding Fig. 7), at high levels of cognitive load TBRS predicts more forgetting with more distractor operations, and this was not observed when distractors were all identical, despite the fact that the experiments implemented a high level of cognitive load. Second, TBRS has no straightforward way of explaining that the effect of number of distractors is modulated by the

similarity between distractors in a burst because TBRS does not care about what is processed in a processing burst, only how long it captures attention.

TBRS* might account for those findings by making the additional assumption that speaking three different words captures attention for a larger proportion of the available time than speaking the same word three times in a row. With Simulation 7 we applied TBRS* to the four conditions of Experiment 3 in Lewandowsky et al. (2010a): After each of five consonants of the memory list, people had to read aloud (1) no words, (2) a single word, which was the same across the whole trial, (3) four times the same word, which again was the same for all bursts within a trial, or (4) three different words, chosen at random for every burst. Participants were instructed to speak the words rapidly and without pausing; as soon as they finished speaking, the experimenter triggered the next display. The mean times for speaking all the words in a burst were 1.15 s for a single word per burst, 2.45 s for four identical words, and 2.42 s for three different words.

For Simulation 7 we made the following assumptions about how these times are divided up: At the beginning of each burst, attention is captured for 0.4 s to switch from encoding a letter to reading words; this estimate is based on unpublished experiments from our laboratory that provide an estimate for switching costs between encoding and processing tasks in a complex span paradigm. In agreement with Liefooghe et al. (2008), this task-switching component is assumed to occupy the attentional bottleneck. Reading a word, preparing the speech plan for it, and initiating speech is assumed to capture attention for 0.3 s per word for a new word.[4] For repeated words we assumed a reduced duration of attentional capture. After some exploration we found that the most extreme assumption—namely, that repeated words do not capture attention at all—results in the best approximation of the mean accuracies of the four conditions; we therefore set operation duration (t_a) for repeated words to zero.

The resulting time schedule for simulation of the experimental conditions is as follows: In the single-distractor condition, 0.7 s of the 1.15 s measured duration is needed for task switching and word reading, this leaves 1.15 – (0.4 + 0.3) = 0.45 s during which the word is spoken while attention is free to refresh memory items. We assume that the same time parameters apply to the first word of the multiple-word bursts. For the constant distractor condition, this means that speaking the first word takes 1.15 s (of which 0.45 s are free time available for refreshing), leaving 2.45 – 1.15 = 1.30 s for speaking the remaining three words. Because repeated words do not engage the bottleneck, this time is assumed to be entirely free for refreshing. In the changing-distractor condition we again assigned 1.15 s to speaking of the first word, of which 0.45 s are free time. The remaining 1.27 s are evenly divided between the remaining two words, so that for each word there were 0.3 s during which the bottleneck is occupied, followed by 0.34 s of free time during which memory items can be refreshed.

Simulation 7 used the same parameter as Simulation 1, with the exception of the noise parameter (σ), which we raised to 0.05 to move overall accuracy into the range of the empirical data. The results are shown in the top panel of Fig. 21; the bottom panel shows the experimental data. At the level of main effects, TBRS* successfully predicts the relative difficulty of the four conditions: Predicted performance is best for the control condition without distractors, and much worse for all three conditions with distractors. Memory with four identical distractors is hardly distinguishable from overall memory with a single distractor. With three changing distractors performance is predicted to be worst.

However, when the results are considered at the level of serial position, the predictions differ considerably from the data. TBRS* predicts that the manipulations of distractor bursts strongly interact with serial position. The first list position is predicted to be largely immune to distractor activity, and the distractor effects increase toward the end of the list. No such interaction was apparent in the experimental data. The reason for the predicted interaction is the cumulative refreshing schedule. Because refreshing starts over at the first list item after every interruption, the first list item is guaranteed to be refreshed whenever there is any amount of free time, but items later in a list have an increasingly smaller chance of being refreshed before the next interruption. This analysis raises the question whether a different refreshing schedule could eliminate the interaction of the distractor manipulation with serial position. We re-ran the simulation with the variant of cumulative refreshing that continues after each operation and jumps back to the start of the list only after a new item is encoded. This was the only other refreshing scheme that we found to generate reasonable predictions of the benchmark findings. With this refreshing scheme, the predictions were largely the same as those in Fig. 21, with the exception that a recency effect was also obtained for the different-distractor condition, thus rendering overall accuracy in that condition very close to that of the identical-distractor

[4] These times for attentional capture by each new word include the costs of switching away from word reading to refreshing, and back from refreshing to word reading; we have no empirical estimates for these switching costs, and we assume here without further justification that they are comparatively short.

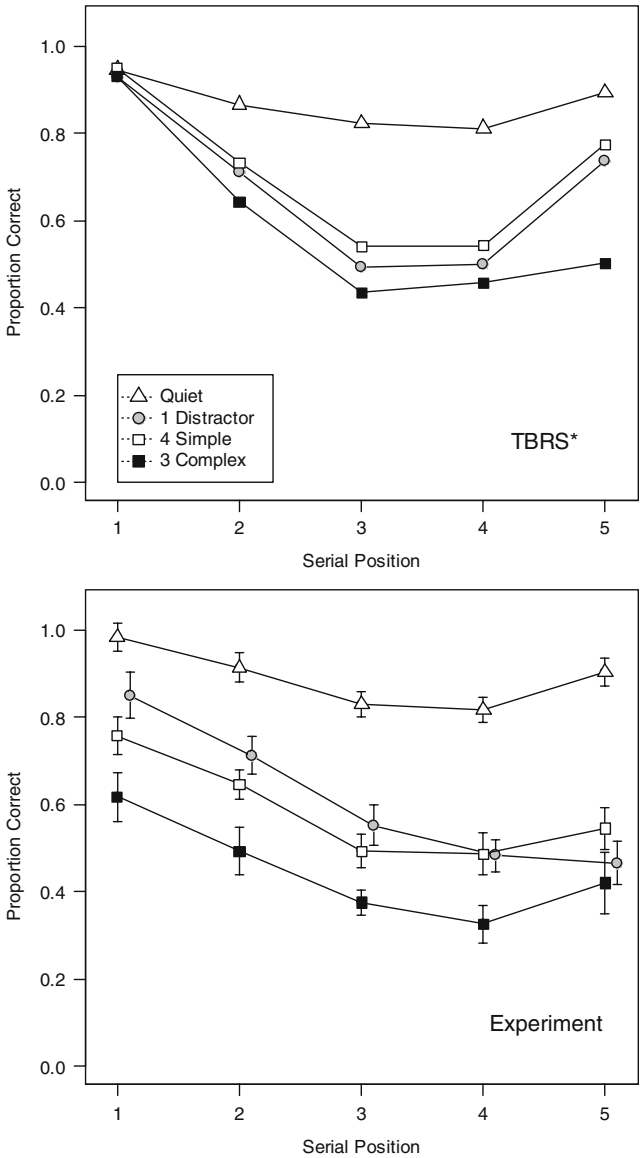

Fig. 21 Model predictions (*top*) and data (*bottom*) from Experiment 3 of Lewandowsky et al. (2010a). "Quiet" refers to the simple-span baseline without articulation; "1 Distractor" denotes the condition with a single word to be read after each item; "4 Simple" denotes reading of four identical words, and "3 Complex" denotes reading of three different words after each item. *Error bars* in the data are within-subject standard errors

condition. The model still predicted a strong interaction of distractor condition with serial position, with hardly any distractor effect on the first position.

In sum, Simulation 7 has shown that, with some auxiliary assumptions about the engagement of the attentional bottleneck during distractor articulation, TBRS* can meet the challenge raised by the findings of Lewandowsky et al. (2010a) at an aggregate level of analysis; that is, at the level of mean performance in each condition. However, as soon as the serial position curves are also considered, the misfit between the model's predictions and the data becomes apparent irrespective of the particular refreshing schedule being used. Although the mismatch between data and predictions challenges the instantiation just presented, it also enhances our confidence about the theory's testability.

It remains for future research to examine the plausibility of the added assumptions that are necessary for TBRS* to handle the data at the level of overall accuracy. The assumption we made here, that repeated utterance of the same word does not recruit the attentional bottleneck at all, contradicts the assumption we had to make in Simulation 2, where repeated utterance of "ba" was assumed to capture attention for about 200 ms every time. At present, we have no independent estimate of the duration of bottleneck engagement during articulation; this is an issue in need of empirical clarification.

General discussion

We presented a computational model of memory performance in the complex span paradigm and related tasks. The model instantiates the assumptions of the TBRS, namely, that memory representations inexorably decay unless refreshed by an attentional bottleneck and that the same bottleneck is also required to perform the processing task. To our knowledge, this represents the first computational model explaining a broad set of benchmark findings with the complex span task, a paradigm central to most working-memory research. Before we discuss the theoretical implications of our work, we briefly review the limitations of our model.

Limitations

The most important limitation of TBRS* is that, by using localist representations, it does not specify similarity relations between the memoranda. It follows that TBRS* cannot handle any of the similarity effects that are known to occur in short term and working memory. This limitation has several ramifications.

We have already shown that TBRS* has difficulty with data that reveal an effect of similarity between distractors: Lewandowsky et al. (2010a) showed that increasing the number of distractors in between study items causes additional forgetting only when each distractor is novel; that is, when distractors are dissimilar from each other. When distractors are identical, no effect of the number of distractors is observed. TBRS* could handle those effects at the aggregate level only by making auxiliary assumptions, namely that articulation of an identical word captures attention for a shorter duration than articulation of a novel word. Even then, the data of Lewandowsky et al. (2010a) remain a challenge for TBRS* because it fails to predict the

correct pattern of interference effects across serial position. It remains to be seen whether introducing a representation that captures inter-item similarity to the model enables a better approximation to those data.

A proper representation of inter-item similarities is also necessary to address the effects of phonological similarity within the list (i.e., among memoranda), which are observed in both simple serial recall and complex span (Tehan, Hendry, & Kocinski, 2001), and the effects of similarity between distractors and memoranda. Concerning the latter, one recent study that used a complex span paradigm found that similarity between memoranda and distractors had little, if any, effect on recall performance (Oberauer, 2009). Whatever small effects were present in the data resulted from feature overlap rather than similarity—that is, if each distractor shared a phonological feature of the preceding memorandum (without however being phonologically similar to it), then recall was slightly impaired. It remains to be seen how those effects can be accommodated; at present, it would be premature to explore potential alternatives within TBRS* because the verbal theory has to date remained mute on any matters relating to similarity.

Another limitation of TBRS* is that it only incorporates one refreshing mechanism. Recently, the verbal version of the TBRS has been extended to incorporate another mechanism by which memory can be restored, namely conventional articulatory rehearsal. Camos, Lagner, and Barrouillet (2009) argued that both types of restoration mechanisms operate in the complex span in an additive and independent manner.

Relationship to other theories

Before we turn to the implications of our modeling, we briefly place our work into a broader theoretical and empirical context. TBRS is intended as a theory of working memory, with rapidly decaying memory traces that support only temporary maintenance, as opposed to much more permanent long-term memory. At the same time, it must be noted that the complex span task shares a number of features with the continuous distractor task that has been commonly regarded as reflecting recall from long-term memory (Bjork & Whitten, 1974). In both tasks, memoranda are separated by distracting activity, and they thus both involve the executive control processes required for task switching. One difference between the paradigms is that the continuous distractor paradigm requests free rather than serial recall, but there have been several recent suggestions that free recall and serial recall are in many regards more similar than had been commonly assumed (Bhatarah, Ward, Smith, & Hayes, 2009; Bhatarah, Ward, & Tan, 2008).

It is not surprising, therefore, that TBRS* also shares theoretical features in common with theories in the long-term memory arena. For example, the notion of temporal-context driven retrieval is shared by several theories in long-term memory, such as the Temporal Context Model (Howard, Fotedar, Datey, & Hasselmo, 2005; Howard & Kahana, 2002) and the SAM model (Gillund & Shiffrin, 1984; Mensink & Raaijmakers, 1988) In the two-store model of Davelaar, Goshen-Gottstein, Ashkenazi, Haarmann, and Usher (2005), a context representation is used for the long-term store, but not for the short-term buffer, which is based entirely on temporary activation of representations. As we detail in Appendix A, we have attempted an implementation of the TBRS in terms of an activation-based buffer that maintains serial order through a primacy gradient, but that attempt failed for principled reasons. Therefore, forming content-context associations appears to be a necessary feature of a successful implementation of the TBRS model for complex span. If the distinction between short-term or working memory on the one hand and long-term memory on the other is cast in terms of temporary activation versus associative learning, as in the model of Davelaar et al. (2005) and many classical dual-store theories (Atkinson & Shiffrin, 1968; Gillund & Shiffrin, 1984), then TBRS* looks more like the long-term than the short-term store.

There are, however, also models of short-term memory utilizing content-context associations to maintain serial order (Burgess & Hitch, 1999; Henson, 1998b). TBRS* can be interpreted as a model in that tradition. The distinction between working and long-term memory could then be seen as the difference between associations that can be formed rapidly and decay rapidly, and others that require more incremental learning and are more permanent, like the "fast weights" and "slow weights" in the model of Burgess and Hitch (1999).

Theoretical implications

The implications of our work are readily summarized. We have shown that there is at least one computational model of the TBRS that "works" in that it handles the core data on which the theory is based: Our model (1) generates roughly linear span-over-load functions; (2) produces only a negligible effect when the number of operations is increased from 4 to 8, (3) produces serial-position curves, transposition gradients, and item error patterns in good agreement with the data. In addition, we showed that TBRS* can handle some developmental data reported by Barrouillet et al. (2009), and data with the Brown-Peterson paradigm by Liefooghe et al. (2008), suggesting that our instantiation has considerable breadth. We identified some limits to that breadth when we applied TBRS* to recent experiments that manipulated the nature of intra-list distractors (Lewandowsky et al., 2010a) and found that it could handle the data only with some auxiliary

assumptions and then only at an aggregate level, without being able to reproduce the pattern of serial position effects.

What can we learn from our efforts? First, we learned something about the TBRS theory, and more generally about theories invoking decay and rehearsal or refreshing as core mechanisms of working memory. Implementing a theory as a computational model is an investigation of what the assumptions of the theory imply. It is by no means guaranteed that the theory, when implemented as a model, behaves as advertised. The TBRS largely does, with one exception: At high levels of cognitive load the model necessarily predicts worse recall when the number of distractors is increased from one to a few (though the effect eventually levels off after several distractors). Thus, the cognitive load equation of Barrouillet et al. (2004) is a good approximation to what the assumptions of the TBRS imply, but it misses at least one detail. Our implementation of TBRS offers a tool to generate more adequate and more detailed predictions.

Second, we learned that in the context of a model of serial recall, decay operates in a way very different from common intuition. Forgetting is not simply a result of representations decaying below a threshold. Rather, recall accuracy is determined by both absolute and relative memory strength after cueing, which in turn results from the complex interplay of decay with sequential encoding, refreshing, and retrieval. This insight should be a warning to those who are attracted to decay-based theories because they appear simple.

Third, we discovered something about the representations that are required to maintain order information in working memory. We showed that a primacy gradient is inadequate (see Appendix A) and that positional markers must be used to represent order. This outcome is consonant with previous results (e.g., Farrell & Lewandowsky, 2004), and it provides a general constraint on other possible models of complex span performance. Our analysis in Appendix A applies not only to the TBRS, but to *any* model that combines a primacy gradient with the presence of rehearsal or refreshing.

Fourth, we discovered that only some, but by no means all, schedules of rehearsal or refreshing provide the benefits to memory that are promised by numerous verbal models, including the TBRS. This outcome has considerable implications beyond the complex span data discussed here, and it extends an earlier analysis (Oberauer & Lewandowsky, 2008) that likewise found rehearsal to be surprisingly ineffective in many circumstances. Skepticism may therefore be advisable when verbal theories purport to explain phenomena via rehearsal; this explanation may or may not work when implemented in detail.

In this article, we showed that TBRS* can handle many existing results, without however providing quantitative fits to the data. We believe that a quantitative evaluation of TBRS* would be premature at this point. A quantitative evaluation of TBRS* must await the development of competing quantitative theories. At the moment, those alternative theories are still under development; for example, the SOB model (Farrell & Lewandowsky, 2002), is currently being extended to handle results from the complex span paradigm. It is only once those promising beginnings have turned into robust theories that a quantitative assessment of TBRS* becomes meaningful—indeed, it then becomes mandatory.

Conclusion and outlook

Can we now safely conclude that performance in the complex span task arises from the balance of decay and rehearsal, exactly as predicted by the TBRS theory? Not necessarily: The existence of TBRS* is a necessary requirement for this to be the case, but it is far from sufficient. In fact, we believe that one of its core assumptions, namely that decay causes forgetting in working memory, is wrong (Lewandowsky, Oberauer, & Brown, 2009). But then, as George Box famously noted, "all models are wrong, but some models are useful" (Box & Draper, 1987, p. 424). We believe that the TBRS model of Barrouillet and colleagues has already proven to be immensely useful in motivating many insightful experiments with the complex span paradigm and in providing a comparatively precise theoretical formulation of the underlying processes. Likewise, we believe that our computational implementation of the TBRS model is a useful model because it represents a first step toward concise modeling of the cognitive processes in an important class of working-memory tasks.

Author Note This work was facilitated by a grant from the Economic and Social Research Council (ESRC), grant RES-062-23-1199 to the first author, as well as a Discovery Grant from the Australian Research Council (ARC) to both authors, a Linkage International Grant from the ARC to both authors and Simon Farrell and Gordon Brown, and an Australian Professorial Fellowship to the second author. We thank Pierre Barrouillet and two anonymous reviewers for valuable comments on an earlier version of this article.

Appendix A: Why the TBRS Cannot Be Implemented with a Primacy Gradient

A primacy gradient codes serial order by activating the representations of list items at encoding to a decreasing extent across serial positions. Recall in the order of presentation is accomplished by competitive cueing: At any point, the most active item is recalled and recall is followed by response suppression, that is, deactivation of the selected item. Thus, at the outset the first item is most active and therefore is recalled before being suppressed. The suppression renders the second list item the most active

one (unless random noise interferes), such that the next item chosen for recall is the second list item, and so on.

The reason why a primacy gradient is not suited for representing order in the complex span paradigm is that it places strong constraints on possible mechanisms of rehearsal or refreshing. Page and Norris (1998) combined a primacy gradient with the assumption of decay and rehearsal in a model of short-term memory, the Primacy Model. Although their description of rehearsal in the Primacy Model is rather sketchy (for a critique see Oberauer & Lewandowsky, 2008), it is the most explicit published specification of rehearsal (or refreshing) in the context of a primacy-gradient model. [Although Page and Norris (1998) consider only articulatory rehearsal, the same argument applies to attention-based refreshing.] Rehearsal is assumed to be cumulative; that is, after each presented item the system rehearses all items encoded so far in their order of presentation. Rehearsal of each item consists of retrieving it and encoding it again, thereby building a new primacy gradient (i.e., activating localist representations of the retrieved items in a second copy of the bank of units representing the items). After the whole list presented so far has been rehearsed, the new primacy gradient, which is relatively fresh, replaces the old, decayed one.

This scheme is very constraining because items can be rehearsed only if there is enough time to rehearse the whole list encoded so far. Rehearsal of only part of the list is not possible because it jeopardizes the integrity of the primacy gradient. Imagine, for example, that the partial list ABCD has been encoded. Activation decreases from A to D with values 1, 0.9, 0.8, and 0.7, respectively (the numbers are arbitrary). Over time this primacy gradient decays to activation levels of about 0.8, 0.7, 0.6, and 0.5, respectively. Assume that now a small time window opens that allows rehearsing of just two items. Using cumulative rehearsal, items A and B could be rehearsed. Following this partial rehearsal, two options exist for the model. One option is that a new primacy gradient is formed with items A and B that replaces the old gradient – but then, items C and D are lost. An alternative option is that rehearsal boosts the activation level of items in the existing primacy gradient. This assumption creates two problems. One is that rehearsal of an item involves retrieval of that item, and in any primacy-gradient based model, retrieval must be followed by suppression of the retrieved item. Suppression of an item, however, is the opposite of what rehearsal or refreshing is meant to achieve. So the modeler faces a dilemma: Either the retrieved item is suppressed, in which case it cannot be refreshed, or the item's activation is boosted, in which case rehearsal can never move beyond the first item because the second item cannot be retrieved.

Even if this problem could be overcome, refreshing by boosting items in the existing primacy gradient leads to another problem. Assume, again, that a small time window allows refreshing of only the first two of four items, so that the activation of A and B in the existing gradient is boosted, while that of C and D continues to decay. Now the next item E is presented, and the question arises how much it should be activated. One option is to determine the activation strength of E relative to the current activation level of the first list item, so that E's activation is what it would be according to a standard primacy gradient that originates from the current activation level—after a rehearsal boost—of the first item. However, in that case the activation assigned to E would surpass the current activation level of C and D—which have not been rehearsed—and would therefore perturb the order coded by the primacy gradient. The other option is to calculate the activation level for E by continuing the gradient down from the current activation level of the immediately preceding non-rehearsed item, D. In that case, E would receive only an activation of 0.3. This option would maintain the primacy gradient, but at the cost that the weakest current activation level sets a ceiling for the encoding of each additional item, and that ceiling continues to decay. Therefore, partial rehearsal or refreshing is ill suited to effectively counteract decay.

This constraint poses a problem that is particularly severe for the TBRS, which assumes a powerful refreshing mechanism that uses even tiny temporal gaps in between two processing steps for refreshing (see Appendix B for a demonstration of how powerful refreshing must be). Under conditions of even moderate cognitive load, the attentional system must switch fairly often between working on the processing task and refreshing, and the time spent on refreshing between presentation of one item and the next will rarely suffice to refresh more than two or three items. Therefore, a primacy gradient based implementation of the TBRS has the choice between two unattractive options. Either it refreshes only when it has time to refresh the whole list encoded at each point in time, in which case it will hardly have enough time to refresh at all once more than the first two or three items have been encoded. Or else the system engages in partial refreshing, which is fairly ineffective for the reasons outlined above.

We implemented the TBRS with a primacy gradient, exploring several refreshing mechanisms and refreshing schedules, but were unable to obtain any reasonable approximation to the data pattern implied by the four key findings mentioned above. We therefore turned our attention to positional coding.

Appendix B: Exploring the Parameter Space of TBRS^0

In a series of simulations using the design of Simulations 1 and 3, we varied the values of four parameters that, in our

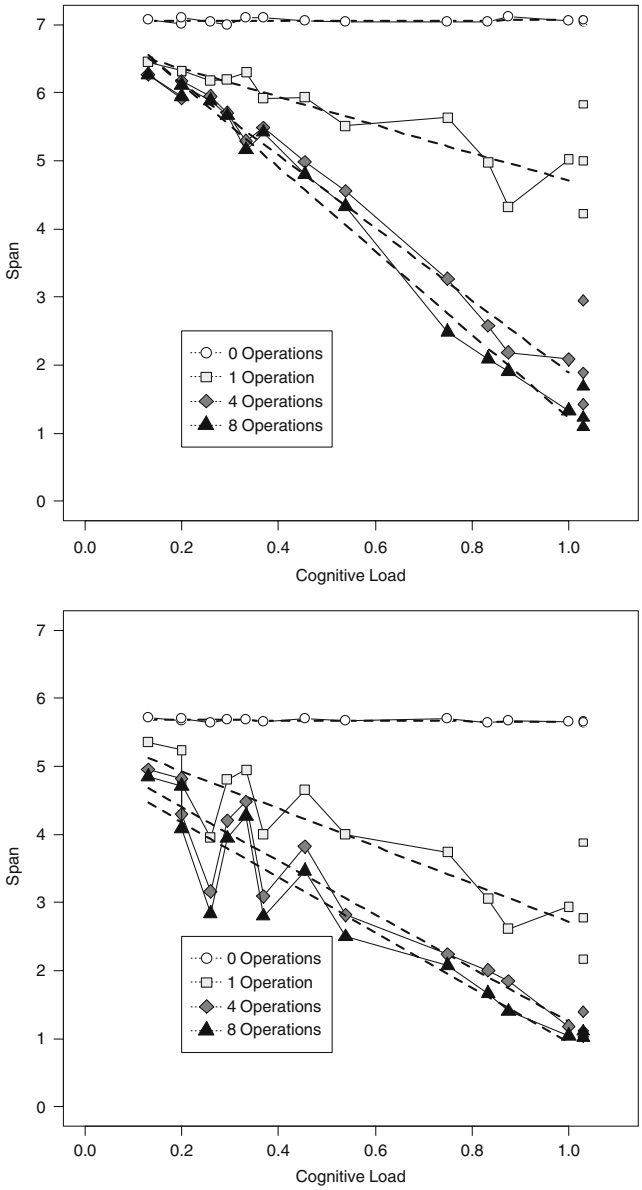

Fig. 22 Span-over-load plots for TBRS^0 with decay rate D = 0.3 and encoding rate R = 4 (*top panel*), and with D = 0.7 and R = 8 (*bottom panel*); all other parameters as in Simulation 3

experience with the model, have the largest impact on its predictions. We completely crossed three levels of decay rate D (0.3, 0.5, 0.7), three levels of mean processing rate R (4, 6, 8), three levels of standard deviation of processing rate s (0.5, 1, 2), and five levels of the duration of a refreshing step T_r (50, 80, 100, 150, 200 ms). The results can be summarized as follows.

Unsurprisingly, increasing decay reduces overall performance. Increasing mean processing rate improved overall performance. Both manipulations changed the pattern of effects of cognitive load and number of operations on performance relatively little; they mainly shifted the pattern up and down. One noticeable change in the data pattern was that with increasing decay rate, and also with increasing processing rate, the non-monotonicity in the performance-over-load curve became more exaggerated. This can be seen in Fig. 22, which shows span-over-load plots for two new combinations of D and R that generate roughly the same overall level of performance as the values of Simulations 1 and 3 (which both used D = 0.5, R = 6). With a lower decay rate and a lower processing rate, as shown in the top panel of Fig. 22, the non-monotonicity is smoothened to some degree, whereas it becomes more pronounced when decay rate and processing rate are both

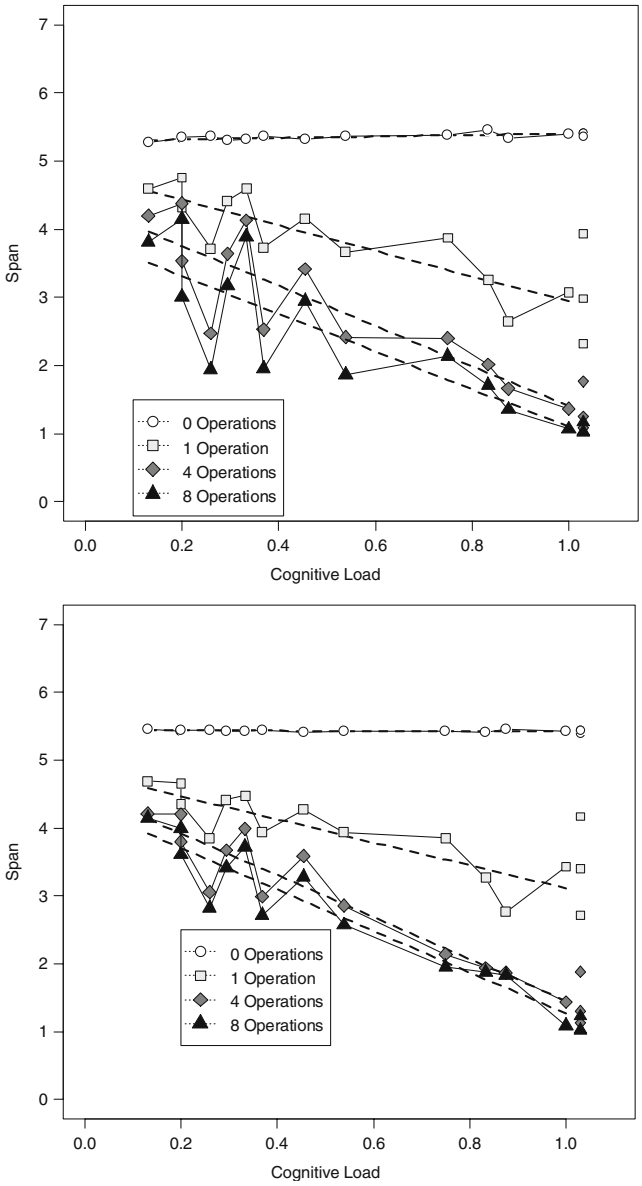

Fig. 23 Span-over-load plots for TBRS^0 with standard deviation of rate s = 2 (*top panel*) and with refreshing duration = 150 ms (*bottom panel*); all other parameters as in Simulation 3

increased, as shown in the bottom panel. The reason for the increased non-monotonicity is that a high decay rate cannot be fully compensated by a high processing rate (implying a high rate of refreshing). As decay rate is increased, items become increasingly likely to become irretrievable, and if the correct item cannot be retrieved, no refreshing, no matter how strong, can save it from obliteration. This is why in the bottom panel of Fig. 22 span values are particularly low at points on the cognitive load axis that reflect long operation durations, which imply long periods of decay.

A monotonic span-over-load function can be regarded as a desirable feature because in none of the experiments manipulating cognitive load so far has any systematic deviation from monotonicity been observed. This consideration makes a combination of relatively small values for *D* and *R* attractive. Against it, however, stands the observation that with a small decay rate and a slow processing rate the effect of the number of operations increases. This is visible in the comparison between the two panels of Fig. 22, and between the top panel of Fig. 22 and Fig. 17. The reason for this observation is that the equilibrium between decay and refreshing is reached faster when decay rate and refreshing rate are both large. Rapid decay combined with strong refreshing means that memory strength moves quickly down (during an operation) and up again (during free time used for refreshing), thus approaching the equilibrium quickly, after relatively few operations, so that additional operations after the first one or two have little effect. With slow decay and weak refreshing, in contrast, the strength level approaches equilibrium in a more leisurely pace, so that increasing the number of operations has an effect even beyond the first one or two operations. This is illustrated in the bottom panels of Fig. 7.

A small effect of the number of operations is a desirable feature of the model because it is supported by the data (Barrouillet et al., 2004; Oberauer & Lewandowsky, 2008), and this militates against the choice of small values for *D* and *R*. We believe that the values chosen for Simulations 1 and 3 are a good compromise between avoiding non-monotonicity in the performance-over-load curve and minimizing the effect of number of operations predicted by the model.

We next look at the effect of varying the standard deviation of processing rate. Decreasing *s* to 0.5 had hardly any discernable effect, apart from a small increase in overall performance. The effect of an increase of *s* to 2 is shown in the top panel of Fig. 23. In addition to dragging down performance, it led again to an exaggeration of the non-monotonicity in the span-over-load curve. The reason for this is that with high variability in *R*, some operation durations are bound to be very long, which again increases the risk for items to become irrecoverable, and this happens particularly often when mean operation duration is long, thus affecting particularly cognitive load levels that combine long operation durations with long free times. We conclude that, whereas values of *s* below 1 are viable, values much above 1 result in undesirable model behavior.

The bottom panel of Fig. 23 shows the effect of increasing the duration of individual refreshing steps to 150 ms. Relative to the value chosen in Simulation 1 (80 ms), increasing this parameter results in three changes: First, performance overall suffers; second, we observed once again an exaggeration of non-monotonicity, because performance suffers particularly for those levels of cognitive load that reflect long operations followed by long free-time periods. Third, the span-over-load curve flattens because performance is particularly depressed at low levels of cognitive load. All three effects can be traced to a weakening of the benefit of refreshing: As the duration of refreshing steps increases, the benefit of free time becomes smaller. The reason for this is that, as the duration of each refreshing step increases, refreshing is distributed less evenly across all items. Within a given window of free time, fewer items can be refreshed, and these items receive relatively strong boosts of memory strength because the amount of boosting is a function of time spent on re-encoding a refreshed item. As a consequence, memory strength becomes increasingly uneven across items as refreshing is concentrated on a smaller subset of items (primarily those at the beginning of the list). We conclude that it is essential for the TBRS to work that refreshing is assumed to be a very rapid process. Whereas it would be difficult to assume such a fast rate for articulatory rehearsal, it is conceivable that items are refreshed at that rate by deploying attention to them.

We conclude from these results that the combination of parameter values chosen for Simulations 1 and 3 is not only sufficient for generating predictions in line with existing data, but also – within certain bounds – necessary for a good approximation to the existing data.

References

Ackerman, P. L., Beier, M. E., & Boyle, M. O. (2005). Working memory and intelligence: The same or different constructs? *Psychological Bulletin, 131*, 30–60.

Atkinson, R. C., & Shiffrin, R. M. (1968). Human memory: A proposed system and its control processes. In K. W. Spence & J. T. Spence (Eds.), *The psychology of learning and motivation: Advances in research and theory* (Vol. 2, pp. 90–195). New York: Academic Press.

Baddeley, A. D. (1986). *Working memory.* Oxford: Clarendon Press.

Barrouillet, P., & Camos, V. (2001). Developmental increase in working memory span: Resource sharing or temporal decay? *Journal of Memory and Language, 45*, 1–20.

Barrouillet, P., Bernardin, S., & Camos, V. (2004). Time constraints and resource sharing in adults' working memory spans. *Journal of Experimental Psychology: General, 133*, 83–100.

Barrouillet, P., Bernardin, S., Portrat, S., Vergauwe, E., & Camos, V. (2007). Time and cognitive load in working memory. *Journal of Experimental Psychology. Learning, Memory, and Cognition, 33*, 570–585.

Barrouillet, P., Gavens, N., Vergauwe, E., Gaillard, V., & Camos, V. (2009). Working memory span development: A time-based resource-sharing model account. *Developmental Psychology, 45*, 477–490.

Bhatarah, P., Ward, G., Smith, J., & Hayes, L. (2009). Examining the relationship between free recall and immediate serial recall: Similar patterns of rehearsal and similar effects of word length, presentation rate, and articulatory suppression. *Memory & Cognition, 37*, 689–713.

Bhatarah, P., Ward, G., & Tan, L. (2008). Examining the relationship between free recall and immediate serial recall: The serial nature of recall and the effect of test expectancy. *Memory & Cognition, 36*, 20–34.

Bjork, R. A., & Whitten, W. B. (1974). Recency-sensitive retrieval processes in long-term free recall. *Cognitive Psychology, 6*, 173–189.

Box, G. E. P., & Draper, N. R. (1987). *Empirical model building and response surfaces*. New York: Wiley.

Brown, J. (1958). Some tests of the decay theory of immediate memory. *The Quarterly Journal of Experimental Psychology, 10*, 12–21.

Brown, S., & Heathcote, A. (2005). A ballistic model of choice response time. *Psychological Review, 112*, 117–128.

Brown, G. D. A., Neath, I., & Chater, N. (2007). A ratio model of scale-invariant memory and identification. *Psychological Review, 114*, 539–576.

Brown, G. D. A., Preece, T., & Hulme, C. (2000). Oscillator-based memory for serial order. *Psychological Review, 107*, 127–181.

Bunting, M. F., Cowan, N., & Saults, J. S. (2006). How does running memory span work? *The Quarterly Journal of Experimental Psychology, 59*, 1691–1700.

Burgess, N., & Hitch, G. J. (1999). Memory for serial order: A network model of the phonological loop and its timing. *Psychological Review, 106*, 551–581.

Burgess, N., & Hitch, G. J. (2006). A revised model of short-term memory and long-term learning of verbal sequences. *Journal of Memory and Language, 55*, 627–652.

Camos, V., Lagner, P., & Barrouillet, P. (2009). Two maintenance mechanisms of verbal information in working memory. *Journal of Memory and Language, 61*, 457–469.

Conway, A. R. A., & Engle, R. W. (1996). Individual differences in working memory capacity: More evidence for a general capacity theory. *Memory, 4*, 577–590.

Conway, A. R. A., Kane, M. J., Bunting, M. F., Hambrick, D. Z., Wilhelm, O., & Engle, R. W. (2005). Working memory span tasks: A methodological review and user's guide. *Psychonomic Bulletin & Review, 12*, 769–786.

Conway, A. R. A., Kane, M. J., & Engle, R. W. (2003). Working memory capacity and its relation to general intelligence. *Trends in Cognitive Sciences, 7*, 547–552.

Cowan, N. (1995). *Attention and memory: An integrated framework*. New York: Oxford University Press.

Cowan, N., Saults, J. S., Elliott, E. M., & Moreno, M. V. (2002). Deconfounding serial recall. *Journal of Memory and Language, 46*, 153–177.

Cowan, N., Towse, J. N., Hamilton, Z., Saults, J. S., Elliott, E. M., Lacey, J. F., et al. (2003). Children's working-memory processes: A response-timing analysis. *Journal of Experimental Psychology: General, 132*, 113–132.

Daneman, M., & Carpenter, P. A. (1980). Individual differences in working memory and reading. *Journal of Verbal Learning and Verbal Behavior, 19*, 450–466.

Daneman, M., & Merikle, P. M. (1996). Working memory and language comprehension: A meta-analysis. *Psychonomic Bulletin & Review, 3*, 422–433.

Davelaar, E. J., Goshen-Gottstein, Y., Ashkenazi, A., Haarmann, H. J., & Usher, M. (2005). The demise of short-term memory revisited: Empirical and computational investigation of recency effects. *Psychological Review, 112*, 3–42.

Dosher, B. A., & Ma, J. J. (1998). Output loss or rehearsal loop? Output-time versus pronunciation-time limits in immediate recall of forgetting-matched materials. *Journal of Experimental Psychology. Learning, Memory, and Cognition, 24*, 316–335.

Ecker, U. K. H., Lewandowsky, S., Oberauer, K., & Chee, A. E. H. (2010). The components of working memory updating: An experimental decomposition and individual differences. *Journal of Experimental Psychology. Learning, Memory, and Cognition, 36*, 170–189.

Farrell, S., & Lewandowsky, S. (2002). An endogenous distributed model of ordering in serial recall. *Psychonomic Bulletin & Review, 9*, 59–79.

Farrell, S., & Lewandowsky, S. (2004). Modelling transposition latencies: Constraints for theories of serial order memory. *Journal of Memory and Language, 51*, 115–135.

Friedman, N. P., & Miyake, A. (2004). The reading span test and its predictive power for reading comprehension ability. *Journal of Memory and Language, 51*, 136–158.

Gillund, G., & Shiffrin, R. M. (1984). A retrieval model for both recognition and recall. *Psychological Review, 91*, 1–67.

Grossberg, S., & Pearson, L. R. (2008). Laminar cortical dynamics of cognitive and motor working memory, sequence learning and performance: Toward a unified theory of how the cerebral cortex works. *Psychological Review, 115*, 677–732.

Grossberg, S., & Stone, G. (1986). Neuronal dynamics of attention switching and temporal order information in short term memory. *Memory & Cognition, 14*, 451–468.

Henson, R. N. A. (1998a). Item repetition in short-term memory: Ranschburg repeated. *Journal of Experimental Psychology. Learning, Memory, and Cognition, 24*, 1162–1181.

Henson, R. N. A. (1998b). Short-term memory for serial order: The Start-End Model. *Cognitive Psychology, 36*, 73–137.

Hintzman, D. L. (1991). Why are formal models useful in psychology? In W. E. Hockley & S. Lewandowsky (Eds.), *Relating theory and data: Essays on human memory in honor of Bennet B. Murdock* (pp. 39–56). Hillsdale: Erlbaum.

Howard, M. W., Fotedar, M. S., Datey, A. V., & Hasselmo, M. E. (2005). The temporal context model in spatial navigation and relational learning: Towards a common explanation of medial temporal lobe function across domains. *Psychological Review, 112*, 75–116.

Howard, M. W., & Kahana, M. J. (2002). A distributed representation of temporal context. *Journal of Mathematical Psychology, 46*, 269–299.

Hudjetz, A., & Oberauer, K. (2007). The effects of processing time and processing rate on forgetting in working memory: Testing four models of the complex span paradigm. *Memory & Cognition, 35*, 1675–1684.

Jolicoeur, P., & Dell'Acqua, R. (1998). The demonstration of short-term consolidation. *Cognitive Psychology, 36*, 138–202.

Kane, M. J., & Engle, R. W. (2000). Working-memory capacity, proactive interference, and divided attention: Limits on long-term memory retrieval. *Journal of Experimental Psychology. Learning, Memory, and Cognition, 26*, 336–358.

Kane, M. J., Hambrick, D. Z., & Conway, A. R. A. (2005). Working memory capacity and fluid intelligence are strongly related

constructs: comment on Ackerman, Beier, and Boyle (2004). *Psychological Bulletin, 131*, 66–71.

Kane, M. J., Hambrick, D. Z., Tuholski, S. W., Wilhelm, O., Payne, T. W., & Engle, R. W. (2004). The generality of working-memory capacity: A latent-variable approach to verbal and visuo-spatial memory span and reasoning. *Journal of Experimental Psychology: General, 133*, 189–217.

Kessler, Y., & Meiran, N. (2006). All updateable objects in working memory are updated whenever any of them are modified: Evidence from the memory updating paradigm. *Journal of Experimental Psychology. Learning, Memory, and Cognition, 32*, 570–585.

Lepine, R., Barrouillet, P., & Camos, V. (2005). What makes working memory spans so predictive of high-level cognition? *Psychonomic Bulletin & Review, 12*, 165–170.

Lewandowsky, S. (1993). The rewards and hazards of computer simulations. *Psychological Science, 4*, 236–243.

Lewandowsky, S., Brown, G. D. A., & Thomas, J. L. (2009). Traveling economically through memory space: Characterizing output order in memory for serial order. *Memory & Cognition, 37*, 181–193.

Lewandowsky, S., Duncan, M., & Brown, G. D. A. (2004). Time does not cause forgetting in short-term serial recall. *Psychonomic Bulletin & Review, 11*, 771–790.

Lewandowsky, S., & Farrell, S. (2008). Short-term memory: New data and a model. In B. H. Ross (Ed.), *The psychology of learning and motivation* (Vol. 49, pp. 1–48). London, UK: Elsevier.

Lewandowsky, S., & Farrell, S. (in press). *Computational modeling in cognition: Principles and practice*. Thousand Oaks, CA: Sage.

Lewandowsky, S., Geiger, S. M., Morrell, D., & Oberauer, K. (2010a). Turning simple span into complex span: Time for decay or interference from distractors? *Journal of Experimental Psychology: Learning, Memory, and Cognition, 36*, 958–978.

Lewandowsky, S., Geiger, S. M., & Oberauer, K. (2008). Interference-based forgetting in short-term memory. *Journal of Memory and Language, 59*, 200–222.

Lewandowsky, S., Oberauer, K., & Brown, G. D. A. (2009). No temporal decay in verbal short-term memory. *Trends in Cognitive Sciences, 13*, 120–126.

Lewandowsky, S., Oberauer, K., Yang, L.-X., & Ecker, U. K. H. (2010b). A working memory test battery for Matlab. *Behavioral Research Methods, 42*, 571–585.

Liefooghe, B., Barrouillet, P., Vandierendonck, A., & Camos, V. (2008). Working memory costs of task switching. *Journal of Experimental Psychology. Learning, Memory, and Cognition, 34*, 478–494.

Maybery, M. T., Parmentier, F. B. R., & Jones, D. M. (2002). Grouping of list items reflected in the timing of recall: Implications for models of serial verbal memory. *Journal of Memory and Language, 47*, 360–385.

Mensink, G.-J., & Raaijmakers, J. G. W. (1988). A model for interference and forgetting. *Psychological Review, 95*, 434–455.

Monsell, S. (2003). Task switching. *Trends in Cognitive Sciences, 7*, 134–140.

Murdock, B. B. (1961). The retention of individual items. *Journal of Experimental Psychology, 62*, 618–625.

Oberauer, K. (2003). Understanding serial position curves in short-term recognition and recall. *Journal of Memory and Language, 49*, 469–483.

Oberauer, K. (2009). Interference between storage and processing in working memory: Feature overwriting, not similarity-based competition. *Memory & Cognition, 37*, 346–357.

Oberauer, K., & Lewandowsky, S. (2008). Forgetting in immediate serial recall: Decay, temporal distinctiveness, or interference? *Psychological Review, 115*, 544–576.

Page, M. P. A., & Norris, D. (1998). The primacy model: A new model of immediate serial recall. *Psychological Review, 105*, 761–781.

Pashler, H. (1994). Dual-task interference in simple tasks: Data and theory. *Psychological Bulletin, 116*, 220–244.

Peterson, L. R., & Peterson, M. J. (1959). Short-term retention of individual verbal items. *Journal of Experimental Psychology, 58*, 193–198.

Pollack, I., Johnson, L. B., & Knaff, P. R. (1959). Running memory span. *Journal of Experimental Psychology, 57*, 137–146.

Portrat, S., Camos, V., & Barrouillet, P. (2008). Working memory in children: a time-constrained functioning similar to adults. *Journal of Experimental Child Psychology.*

Ratcliff, R., & Smith, P. L. (2004). A comparison of sequential sampling models for two-choice reaction time. *Psychological Review, 111*, 333–367.

Raven, J. C., Court, J. H., & Raven, J. (1977). *Raven's progressive matrices and vocabulary scales*. New York: Psychological Corporation.

Raye, C. L., Johnson, M. K., Mitchell, K. J., Greene, E. J., & Johnson, M. R. (2007). Refreshing: A minimal executive function. *Cortex, 43*, 135–145.

Surprenant, A. M., & Neath, I. (2009). *Principles of memory.* New York: Taylor & Francis.

Tehan, G., Hendry, L., & Kocinski, D. (2001). Word length and phonological similarity effects in simple, complex, and delayed serial recall tasks: Implications for working memory. *Memory, 9*, 333–348.

Towse, J. N., Hitch, G. J., & Hutton, U. (2000). On the interpretation of working memory span in adults. *Memory & Cognition, 28*, 341–348.

Turley-Ames, K. J., & Whitfield, M. M. (2003). Strategy training and working memory task performance. *Journal of Memory and Language, 49*, 446–468.

Turner, M. L., & Engle, R. W. (1989). Is working memory capacity task dependent? *Journal of Memory and Language, 28*, 127–154.

Unsworth, N., & Engle, R. W. (2006). A temporal-contextual retrieval account of complex span: An analysis of errors. *Journal of Memory and Language, 54*, 346–362.

Usher, M., & McClelland, J. L. (2001). The time course of perceptual choice: The leaky, competing accumulator model. *Psychological Review, 108*, 550–592.

Vergauwe, E., Barrouillet, P., & Camos, V. (2009). Visual and spatial working memory are not that dissociated after all: A time-based resource-sharing account. *Journal of Experimental Psychology. Learning, Memory, and Cognition, 35*, 1012–1028.